Diss. ETH Nr. 16879

Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme

Abhandlung

zur Erlangung des Titels

Doktor der Technischen Wissenschaften

der

Eidgenössischen Technischen Hochschule Zürich

vorgelegt von

Jens Bathelt

Dipl. Technomathematiker, Universität Duisburg

geboren am 3. Juni 1968

von Dortmund-Hörde, Deutschland

Angenommen auf Antrag von

Prof. Dr. Konrad Wegener, Referent

Prof. Dr. Jürgen Gausemeier, Korreferent

2006

Vorwort

Die vorliegende Dissertation entstand während meiner Zeit als wissenschaftlicher Mitarbeiter und Assistent am Zentrum für Produkt-Entwicklung (ZPE) der ETH Zürich.

Für die vielfältige Unterstützung dieser Arbeit durch Professoren, Kollegen, Studenten und Industriepartner möchte ich mich an dieser Stelle bedanken. Insbesondere hervorzuheben ist Prof. Markus Meier, der mich als Quereinsteiger mit offenen Armen am ZPE empfangen und unterstützt hat. Bis zu seinem tragischen Tod im April 2005 hat er mich als Doktorvater betreut und gefördert, indem er ein inspirierendes Umfeld zwischen Mensch, Industrie und Hochschule schuf. Prof. Konrad Wegener danke ich für die erreichte Kontinuität am ZPE und meiner Forschungsprojekte mit der Industrie. Als Erstgutachter hat er durch konstruktive Kritik die vorliegenden Arbeit bereichert. Prof. Jürgen Gausemeier danke ich für das Interesse an dieser Arbeit, sein wertvolles Feedback zur Dissertation und die Übernahme des Korreferats.

Bei meinen Kollegen bedanke ich mich für das freundschaftliche Arbeitsklima und die interessanten Diskussionen, welche meinen Horizont erweitert haben. Auch ausserhalb der Forschungsarbeit wird mir die mannigfaltige Bereicherung im Privaten in bester Erinnerung bleiben! Dr. Andreas Kunz hat mir einen guten Start in der VR-Gruppe am ZPE ermöglicht. Dr. Stefan Dierssen hat mich in die Prinzipien der Virtuellen Maschine eingeweiht. Christian Bacs scheute kein Risiko und hat meine Forschung als Student, Praktikant und Doktorand bereichert. Dr. Anders Jönsson verdanke ich interessante Herausforderungen beim Aufbau einer virtuellen Maschine an der BTH in Schweden und erfreue mich des kontinuierlichen Austauschs. Josef Meile danke ich herzlich für die Unterstützung bei der ELVAN-Implementation.

Der Kommission für Technologie und Innovation (KTI) gebührt der Dank für die finanzielle Unterstützung meines Forschungsprojektes EVA (Early Virtual mAchine). Die teilnehmenden Industriepartner haben im Rahmen eines offenen, freundschaftlichen und konstruktiven Klimas Praxisbeispiele, Wissen und Erfahrungen aus der Praxis, Steuerungshardware und Entwurfssoftware zur Verfügung gestellt: Erwin Pfister + Heinz Studer (Rieter), Urs Müller + Marc Mouthon (Gritec), Hansruedi Wipf + Martin Triet (Brütsch) und Dr. Stefan Dierssen + Jens Byland (Intelliact). Auch ausserhalb des EVA-Projekts wurde meine Forschung durch die Industrie unterstützt. In unkomplizierter Art und Weise bekam ich wertvolle Industriekontakte sowie Hard- und Software von Hans Menzi (SIEMENS). Jürgen Mewes (Mewes & Partner) danke ich für die bereitgestellte Simulationssoftware.

Zürich, im Oktober 2006 *Jens Bathelt*

„Der Weg ist das Ziel.“

Konfuzius (551 v.Chr. - 479 v.Chr.)

Inhaltsverzeichnis

Abkürzungsverzeichnis VII
Zusammenfassung IX
Abstract XI
1 Einführung 1
1.1 Problemstellung 1
1.2 Zielsetzung und Anforderungen 5
1.3 Aufbau der Arbeit 6
2 Stand der Technik 7
2.1 SPS-gesteuerte mechatronische Systeme 7
2.2 Konstruktion 10
2.3 Steuerungstechnik 15
2.4 Interdisziplinäre Entwicklung 20
2.4.1 Buur 24
2.4.2 Stone 25
2.4.3 Kallenbach 27
2.4.4 Flath 29
2.4.5 Molt 30
2.4.6 O^2MEN / CAMeL 31
2.4.7 Schemebuilder 34
2.4.8 Process Oriented Analysis (POA) 35
2.4.9 Osmers 37
2.4.10 eM-PLC 38
2.5 Handlungsbedarf 40
2.5.1 Beschreibung des interdisziplinären Konzeptes durch eine gemeinsame Sprache für SPS-gesteuerte mechatronische Systeme (A1.1) 40
2.5.2 Reibungsloser Übergang vom gemeinsam erstellten Konzept zum Steuerungs- und Konstruktionsentwurf (A1.2) 41
2.5.3 Verbesserung der Konsistenz interdisziplinär relevanter Daten im domänenspezifischen Entwurf (A1.3) 42
2.5.4 Planmässiges Vorgehen zur Entwicklung SPS-gesteuerter Systeme (A2) 43
2.5.5 Unterstützung der Konstruktion und Steuerungstechnik durch Software (A3) 43
2.5.6 Fazit 44
3 Interdisziplinäre Entwicklung SPS-gesteuerter mechatronischer Systeme 45
3.1 Neue Methoden 46
3.1.1 EFS: Erweiterte Funktionsstruktur 46
3.1.2 EFS zur Ableitung des SFC, der I/O-Liste und der Baugruppen 48
3.1.3 EFS als Brücke zwischen der Konstruktion und der Steuerungstechnik 51
3.2 Adaption des planmässigen Vorgehens 54
3.3 Neue Software zur Unterstützung der Methoden 62

4 Fallstudie ... 73
4.1 Die Kämmmaschine der Firma Rieter im Spinnprozess ... 73
4.2 Erstellung der EFS in ELVAN ... 76
4.3 Nutzung der in ELVAN definierten EFS ... 82
4.3.1 Der Weg zum Konstruktionsentwurf ... 82
4.3.2 Der Weg zum Steuerungsentwurf ... 86
4.3.3 Der Weg zur virtuellen Inbetriebnahme ... 89
5 Abschliessende Betrachtungen ... 91
5.1 Diskussion ... 91
5.2 Ausblick ... 94
5.2.1 Verallgemeinerung der Entwicklungsmethodik ... 94
5.2.2 Rapid Virtual Prototyping ... 94
Anhang: Benutzerhandbuch der Software ELVAN ... 97
Referenzen ... 137

Abkürzungsverzeichnis

AS ..... Ablaufsprache (englisch: SFC)
AWL ..... Anweisungsliste (englisch: IL)
BTH ..... Blekinge Tekniska Högskola (englisch: Blekinge Institute of Technology)
CAD ..... Computer Aided Design
CAMeL ..... Computer-Aided Mechatronics Laboratory
DIN ..... Deutsches Institut für Normung
EFS ..... Erweiterte Funktionsstruktur / extended function structure
ELVAN ..... EarLy Virtual mAchine applicatioN
ETH ..... Eidgenössische Technische Hochschule
EVA ..... Early Virtual mAchine
FBD ..... Function Block Diagram (deutsch: FUP)
FMS ..... Function-means structure
FMT ..... Function-means tree
FS ..... Funktionsstruktur / function structure
FUP ..... Funktionsplan (englisch: FBD)
FOD ..... Function Oriented Design
GRAFCET ..... GRAphe Fonctionnel de Commande Etape Transition
GUI ..... Graphical User Interface
IL ..... Instruction List (deutsch: AWL)
I/O-Liste ..... Liste aller Input- und Outputvariablen
ISO ..... International Organization for Standardization
JT ..... Jupiter Tesselation
KOP ..... Kontaktplan (englisch: LD)
KTI ..... Kommission für Technologie und Innovation
LD ..... Ladder Diagram (deutsch: KOP)
MCAD ..... Mechanical Computer Aided Design
MMI ..... Mensch-Maschine-Interface
O^2MEN ..... Objektorientierten Mechatronik Entwurfs
OMM ..... Objektorientiertes Mechatronikmodell
OPC ..... Open Process Control
PDM ..... Product Data Management

PLC..... Programmable Logic Controller (deutsch: SPS)

PLM..... Product Lifecycle Management

POA..... Process Oriented Analysis

SFC..... Sequential Function Chart (deutsch: AS)

SPS ..... Speicherprogrammierbare Steuerung

ST ..... Structured Text / Strukturierter Text

UML..... Unified Modeling Language

VDI..... Verein Deutscher Ingenieure

VR ..... Virtuelle Realität

VRML ..... Virtual Reality Modeling Language

XML..... Extensible Markup Language

ZPE..... Zentrum für Produkt-Entwicklung

Zusammenfassung

In der Mechatronik stellt das gemeinsame Entwickeln - vor allem domänenübergreifend - eine Herausforderung dar. Traditionell werden SPS-gesteuerte mechatronische Systeme wie z.B. Textil- oder Verpackungsmaschinen zuerst in der mechanischen Konstruktion mittels MCAD entworfen, bevor die SPS in der Steuerungstechnik programmiert wird. Dadurch wird das Potential der Steuerungstechnik anfangs vernachlässigt. Die ungenügende Synchronisation der Domänen führt zusätzlich zu inkonsistenten Entwürfen, wodurch zusätzliche Kosten- und Zeitaufwände generiert werden.

In der vorliegenden Arbeit wird eine Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme vorgestellt, in der die Steuerungstechnik bereits in der frühen Konzeptphase methodisch einbezogen und mit der Konstruktion während des Entwicklungsprozesses synchronisiert wird. Schon bei der Beschreibung der notwendigen Funktionalität des Systems können mit der in dieser Arbeit präsentierten *Erweiterten Funktionsstruktur* (EFS) nicht nur die klassischen Funktionen des elektromechanischen Subsystems der Konstruktion, sondern auch die Ablauflogik der SPS abgebildet werden.

Neben existierenden Methoden zur Modularisierung und Konkretisierung des elektromechanischen Subsystems um den Konstruktionsentwurf im CAD einzuleiten, wird eine neue Methode zur Initialisierung des Entwurfs in der Steuerungstechnik vorgestellt. Diese Methode erlaubt das automatisierte Ableiten eines genormten SPS-Programms ausgehend von der interdisziplinär erstellten EFS.

Änderungen der Aktorik oder Sensorik betreffen sowohl den CAD-Entwurf, als auch das SPS-Programm. Da diese Änderungen in der domänenspezifischen Entwurfssoftware durchgeführt und nicht systematisch der jeweils anderen Domäne kommuniziert werden, kommt es zu inkonsistenten Komponentenentwürfen. Basierend auf der EFS wird ein neuer Ansatz zur Identifikation der interdisziplinären Auswirkungen solcher Änderungen vorgestellt.

Auf Basis der VDI-Richtlinie, die eine Entwicklungsmethodik für allgemeine mechatronische Systeme beschreibt, wird in dieser Arbeit das Erstellen und Nutzen der EFS in ein planmässiges Vorgehen der VDI 2206 über vordefinierte Prozessbausteine eingebettet. Dadurch wird aufge-

zeigt, wie eine Parallelisierung der Steuerungstechnik mit der Konstruktion im Entwicklungsprozess erreicht werden kann. Einer der Prozessbausteine fokussiert dabei auf die effiziente Erstellung virtueller Prototypen. Es werden Arbeitschritte definiert, die zu einer Verknüpfung der SPS-Software mit der CAD-Geometrie über die in der EFS gesammelten Daten zur verwendeten Aktorik und Sensorik führen.

Für den Entwurf SPS-gesteuerter mechatronischer Systeme hat sich das CAD in der Konstruktion und die SPS-Programmierumgebung in der Steuerungstechnik bewährt. Eine Software zur domänenübergreifenden Konzeption fehlt. Das hierfür neu entwickelte Tool ELVAN (EarLy Virtual mAchine applicatioN) unterstützt die neuen Methoden zur interdisziplinären Beschreibung des Konzeptes und fügt sich in die bestehende Softwarelandschaft zur Entwicklung SPS-gesteuerter mechatronischer Systeme ein.

Anhand einer Textilmaschine wird die vorgestellte Entwicklungsmethodik exemplarisch erläutert. Zuerst wird in der Software ELVAN die EFS der Maschine formuliert. Anschliessend wird gezeigt, wie die EFS mit Hilfe von ELVAN für den domänenspezifischen Entwurf genutzt werden kann. Das Ableiten eines virtuellen Prototypen der Textilmaschine aus den Daten der ELVAN, des CAD und der SPS-Programmierumgebung runden diese Fallstudie ab.

Insgesamt zeigt diese Arbeit auf, wie die Steuerungstechnik gemeinsam mit der Konstruktion durch die Verwendung der EFS ein SPS-gesteuertes mechatronisches System effizient und parallel entwickeln kann.

Abstract

The collaboration when developing mechatronic products - in particular across the domain borders - is challenging. Mechatronic systems controlled by a PLC, like textile or packaging machines, are usually designed sequentially. Firstly, the mechanical engineer is working on the machine concept and building the MCAD model, followed by the control engineer implementing the software for the control. The potential of the control software is neglected in the early design phase due to that sequential procedure. In addition, the lacking synchronisation between the domains leads to inconsistencies in the design, implying higher costs and a longer development time.

This work presents a design methodology for mechatronic systems controlled by a PLC, where the control engineers are methodically involved in the early conceptual phase and synchronized with the mechanical engineers throughout the development process. The developed *Extended Function Structure* (EFS) does not only contain the traditional functions of the electromechanical subsystem, but also control sequences of the PLC.

A new method to initialise the domain-specific embodiment design uses the data collected in the EFS. The control engineer can derive a standardised PLC-program from the EFS automatically. It is shown how the mechanical engineer can combine existent methods to derive modules and concretising the electromechanical subsystem with the EFS in order to start CAD modelling.

Changing actuators or sensors has an impact on the PLC-program and the CAD-model. Those changes are carried out either by the control or the mechanical engineer in the domain-specific software and not transmitted systematically to the other domain. An approach based on the EFS shows how to identify the impact caused by changes crossing the domain.

The VDI-guideline 2206 describes a design methodology for mechatronic systems in general. The definition and usage of the EFS is embedded into the structured procedure of the VDI 2206 by the definition of predefined process modules. This shows how to enable concurrent engineering throughout the development process. One of the process modules describes how to setup a virtual prototype efficiently. The defined working steps are leading to a linkage between the

PLC-software and the CAD-geometry by the data from the used actuators and sensors collected in the EFS.

Established tools during the domain-specific embodiment design phase are CAD-Systems and the PLC-programming environment. A tool supporting the common conceptual design phase is missing. ELVAN (EarLy Virtual mAchine applicatioN) is filling that gap. It supports the new methods based on the EFS and harmonises with established design tools to develop mechatronic systems controlled by a PLC.

A case study does exemplify the presented design methodology by the use of an textile machine. Firstly, the EFS for the machine is modelled in ELVAN. Secondly, it is shown how the domain-specific embodiment design can gain by the usage of the EFS defined in ELVAN. Finally, a virtual prototype of the textile machine is derived from the data collected in ELVAN, the CAD-model and the PLC-programming environment.

In summary, this work has shown how the control and the mechanical engineer can collaborate efficiently and concurrently by using the EFS when developing a mechatronic system controlled by a PLC.

1 Einführung

1.1 Problemstellung

In den letzten Jahrzehnten hat die Komplexität der Produkte im Maschinenbau kontinuierlich zugenommen. Rein mechanische Lösungen wurden durch mechatronische Produkte ersetzt, die neben der Mechanik zusätzlich Elektronik und Softwaretechnik beinhalten. Der X-by wire Ansatz (bei dem rein mechanische Teillösungen durch mechatronische Teillösungen ersetzt werden) zeigt, dass auch bei existierenden mechatronischen Produkten der rein mechanische Anteil immer weiter reduziert wird [72].

Der immer grösser werdende Anteil an Software in mechatronischen Produkten schlägt sich auch in der prozentualen Aufteilung der Herstellungskosten nieder. Wie aus Bild 1 von [63] ersichtlich, ist der Softwareanteil der Herstellungskosten mechatronischer Produkte in den letzten Jahrzehnten von weniger als 5% auf 40% gestiegen. Damit erreicht die Steuerungstechnik durch die Entwicklung von Software dasselbe Gewicht wie die Konstruktion mechanischer Komponenten.

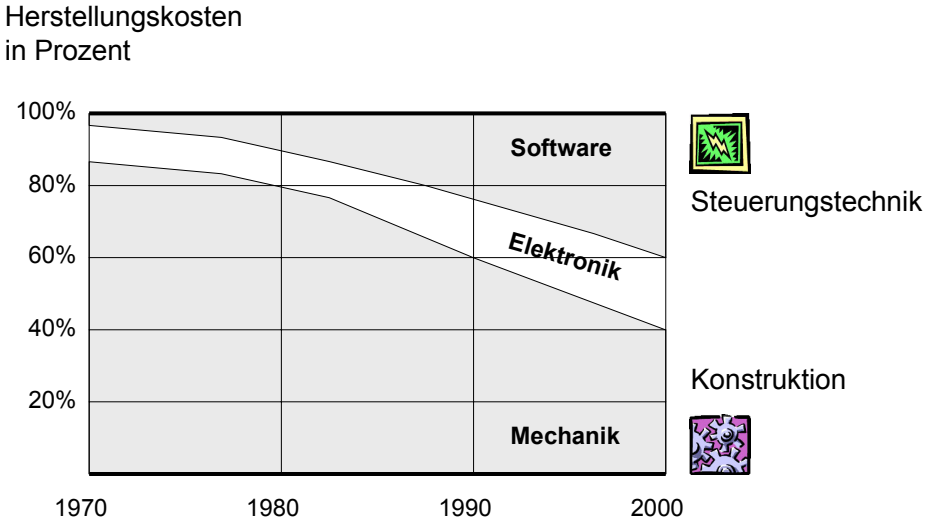


Bild 1: Entwicklung der Produktanteile in der Mechatronik [63]

Die historische Dominanz der Konstruktion spiegelt sich auch heute noch im Entwicklungsprozess mechatronischer Produkte (Bild 2) wieder. Trotz der immer stärkeren Beeinflussung des mechatronischen Produktes durch Software wird die Steuerungstechnik erst spät im Entwicklungsprozess einbezogen.

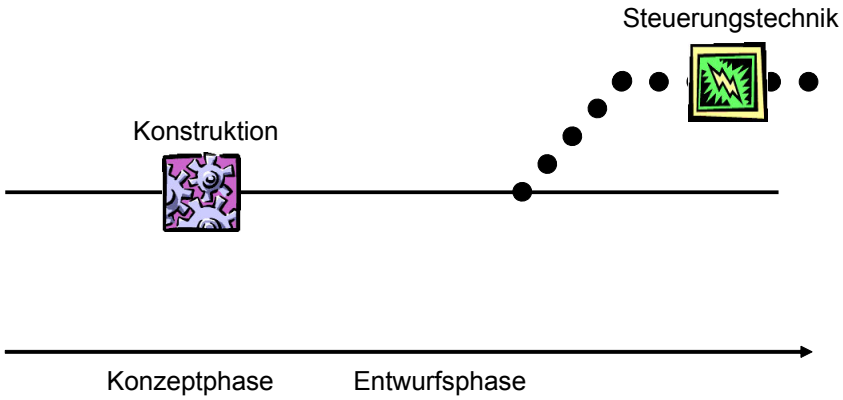


Bild 2: Heutiger Workflow [3]

Traditionell wird in der Konstruktionsabteilung das mechatronische System konzipiert und anschliessend mittels CAD (Computer Aided Design) entworfen. Die Maschinenabläufe werden vom Maschinenhersteller typischerweise unter Verwendung der Sprachen aus der DIN EN 61131-3 [23] implementiert und auf eine Speicherprogrammierbare Steuerung (SPS) geladen. Beispiele für solche SPS-gesteuerten mechatronischen Systeme sind Textilmaschinen [61] und Verpackungsmaschinen [9] wie sie in Bild 3 dargestellt sind.

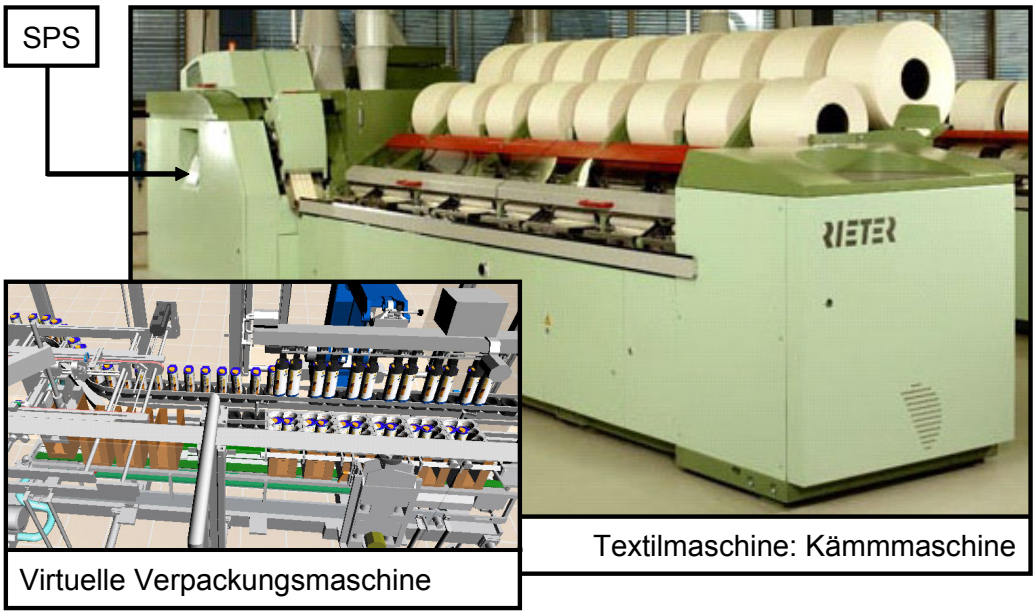


Bild 3: Zwei Beispiele SPS-gesteuerter mechatronischer Systeme

Die SPS ist ein Ablaufsteuerung [79], in der die Ablauflogik programmiert wird. Obwohl dem Konstrukteur die notwendigen Maschinenabläufe schon in der Konzeptphase bewusst sind, startet die Steuerungstechnik mit dem Programmieren der Sequenzen in der Regel erst nach dem Entwurf der Konstrukteure. Dieses sequentielle Vorgehen bei der Entwicklung führt zu teiloptimierten Produkten, die langwierige Iterationen mit kosten- und zeitintensiven Entwicklungsprozessen nach sich ziehen [72]. Die Randbedingungen und das Potential der Steuerungstechnik wird dabei in der Konzeptphase vernachlässigt. Fehler im Konzept, die die Steuerungstechnik betreffen, können während der Konzeptphase nicht entdeckt werden und führen später zu kostspieligen Änderungen in der Konstruktion. Darüber hinaus fehlt eine Synchronisation der interdisziplinären Daten in der Entwurfsphase.

Wird zum Beispiel ein zusätzlicher Sensor zur Überprüfung des Verschlusses einer grossen Klappe aus Stabilitätsgründen im CAD hinzugefügt, so muss dies in der Steuerungstechnik berücksichtigt werden. Bei der reinen Softwareentwicklung würde der Kompiler eine zusätzliche nicht verwendete Variable melden. Die fehlende Integration von CAD und SPS-Programmierumgebung erlaubt solche Konsistenzprüfungen nicht [62].

Analoge Problemstellungen hatten auch die Firmen Rieter Textile Systems [61], Gritec [29], Brütsch [11] und Intelliact [33]. In einem von der KTI [46] geförderten Forschungsprojekt wurde die geschilderte Problematik gemeinsam mit der ETH Zürich und den Industriepartnern angegangen.

Rieter ist ein weltweit tätiges Unternehmen und entwickelt SPS-gesteuerte Textilmaschinen. Für die Entwicklung der SPS-Software und der CAD-Modelle gibt es jeweils die Abteilungen Konstruktion und Steuerungstechnik innerhalb der Firma. Hier ist der Workflow aus Bild 2 mit vielen Zyklen zwischen Konstruktion und Steuerungstechnik anzutreffen. Die Synchronisation - auch global über Standorte hinweg - war bei der Firma Rieter die grösste Motivation an der Teilnahme des Projektes.

Gritec ist Dienstleister in der Domäne Konstruktion und arbeitet bei der Realisation von SPS-gesteuerten mechatronischen Systemen mit externen Partnern zusammen. Hier steht das Problem der gemeinsamen Beschreibungssprache im Vordergrund. Gritec beschreibt alle Maschinenabläufe inklusive der verwendeten Aktoren und Sensoren und deren Variablennamen in MS Excel. Die externen Partner leiten daraus eine eigene Beschreibung der Abläufe ab und vergeben neue SPS-konforme Variablennamen. Die fehlende gemeinsame Konzeptbeschreibung ohne domänenübergreifende Namenskonvention der Baugruppen, Aktoren, Sensoren und SPS-Variablennamen führte zu teuren Inkonsistenzen im Entwurf.

Brütsch ist Dienstleister in der Domäne Steuerungstechnik und wird von Firmen wie z.B. Rieter oder Gritec beauftragt. Neben der reinen Implementation von SPS-Lösungen bietet Brütsch auch die Konzeption und die Inbetriebnahme an. Brütsch erlebt die gespiegelte Variante der von der Firma Gritec geschilderten Probleme. Allerdings hat Brütsch weniger Einfluss auf das gemeinsame Vorgehen mit der Konstruktion, da Brütsch im Gegensatz zu Gritec Auftragnehmer ist.

Intelliact berät und unterstützt Kunden bei der Integration von CAD/PLM-Lösungen und der Virtuellen Maschine [17]. Durch die kommerzielle Umsetzung der Virtuellen Maschine traten bei der virtuellen Inbetriebnahme Inkonsistenzen zu Tage, die ansonsten erst bei der realen Inbetriebnahme aufgefallen wären. Der Wunsch der Kunden, diese domänenübergreifenden Inkonsistenzen möglichst schon im Entwurf zu vermeiden, war die Motivation der Firma

Intelliact diese Entwicklungsproblematik im Rahmen des KTI-Forschungsprojektes gezielt anzugehen.

1.2 Zielsetzung und Anforderungen

Die im letzten Abschnitt geschilderten Probleme führen dazu den Workflow aus Bild 4 anzustreben.

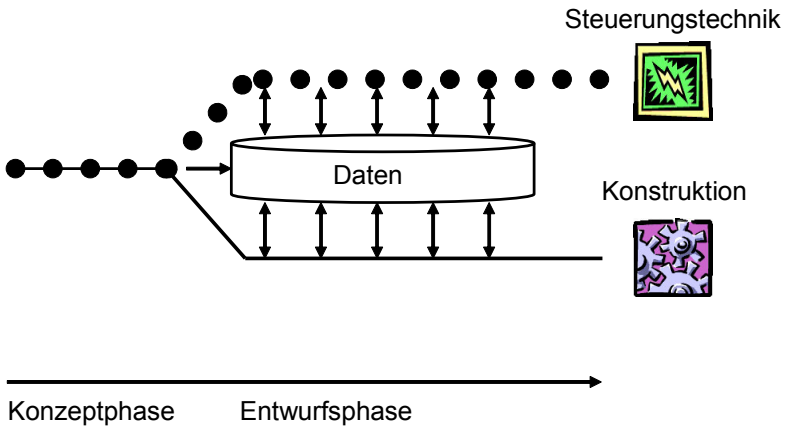


Bild 4: Angestrebter Workflow [3], [4]

Hier wird die Steuerungstechnik schon in der Konzeptphase eingebunden. Anschliessend findet ein strukturierter Übergang in die Entwurfsphase statt, wo die Konzepte innerhalb der Domänen ausgearbeitet werden. Die Steuerungstechnik und die Konstruktion entwerfen nach wie vor das Produkt mit ihren domänenspezifischen Werkzeugen. Zusätzlich findet eine Parallelisierung und Synchronisation zwischen den Domänen statt.

Das Ziel dieser Arbeit ist es, den in Bild 4 gezeigten Workflow durch eine angepasste Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme zu erreichen. Eine Methodik besteht nach Pahl-Beitz [59] aus einem planmässigen Vorgehen unter Einschluss mehrerer Methoden und Hilfsmittel. Im Kontext der in Abschnitt 1.1 dargelegten Problematik lassen sich die Anforderungen (A1)-(A3) an die Problemlösung wie folgt auflisten:

- (A1) *Methoden* zur Verbesserung der interdisziplinären Zusammenarbeit zwischen Konstruktion und Steuerungstechnik:
  - (A1.1) Beschreibung des interdisziplinären Konzeptes durch eine gemeinsame Sprache
  - (A1.2) Reibungsloser Übergang vom gemeinsam erstellten Konzept zum Steuerungs- und Konstruktionsentwurf
  - (A1.3) Verbesserung der Konsistenz interdisziplinär relevanter Daten im domänenspezifischen Entwurf
- (A2) *Planmässiges Vorgehen* zur Entwicklung SPS-gesteuerter Systeme, insbesondere für die Konzept- und Entwurfsphase inklusive der Phasenübergänge
- (A3) Die Methodik soll durch Software als *Hilfsmittel* für die Konstruktion und Steuerungstechnik unterstützt werden.

1.3 Aufbau der Arbeit

Nach der Einführung in die Problemstellung dieser Arbeit und der Formulierung von Anforderungen zur Problemlösung in diesem Kapitel, wird im nächsten Kapitel der Stand der Technik in diesem Kontext analysiert und diskutiert. Zuerst findet eine Beschreibung des prinzipiellen Aufbaus SPS-gesteuerter mechatronischer Systeme statt. Domänenspezifische und domänenübergreifende Ansätze zur Problemlösung folgen. Diese werden abschliessend zur Identifikation des Handlungsbedarfs den Anforderungen gegenübergestellt.

Der in Kapitel 2 aufgezeigte Handlungsbedarf führt zu der in Kapitel 3 vorgestellten Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme. Dabei werden existierende Ansätze wie die VDI Richtlinie 2206 (Abschnitt 2.4) mit neu entwickelten Methoden, neuen Vorgehensweisen und einer neuen Software verknüpft.

Zur Veranschaulichung der in Kapitel 3 erzielten Forschungsresultate werden diese in Kapitel 4 auf eine Textilmaschine angewandt. Es handelt sich dabei um die in Bild 3 dargestellte Kämmmaschine der Firma Rieter. Dabei wird die neu entwickelte Software in Kombination mit existierenden Entwurfswerkzeugen eingesetzt.

Kapitel 5 fasst schlussendlich die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf zukünftigen Forschungsbedarf.

2 Stand der Technik

In diesem Kapitel wird der Stand der Technik in Bezug auf die Entwicklung SPS-gesteuerter mechatronischer Systeme beleuchtet. Zunächst geht der Abschnitt 2.1 auf die Spezialisierung ein, die sich durch die SPS ergibt. Da die Entwicklung SPS-gesteuerter mechatronischer Systeme wesentlich durch die Domänen Konstruktion und Steuerungstechnik geprägt ist [1], [3], [4], wird anschliessend der domänenspezifische Entwurf beschrieben. Typischerweise wird dabei heutzutage zuerst der Konstruktionsentwurf (Abschnitt 2.2) und anschliessend der Steuerungsentwurf (Abschnitt 2.3) mit entsprechenden Rückkopplungen realisiert (Bild 2). Das Ziel einer gemeinsamen parallelen interdisziplinären Entwicklung mechatronischer Systeme (Bild 4) ist nicht neu. Ansätze dazu werden in Abschnitt 2.4 präsentiert. Im Hinblick auf die Ziele der Arbeit liegt dabei der Schwerpunkt auf der interdisziplinären Konzeption inklusive strukturiertem Übergang zu den domänenspezifischen Entwurfswerkzeugen für die Entwicklung SPS-gesteuerter mechatronischer Systeme. Abschliessend werden die Anforderungen aus Abschnitt 1.2 dem Stand der Technik in Abschnitt 2.5 gegenübergestellt um den verbleibenden Handlungsbedarf aufzuzeigen.

2.1 SPS-gesteuerte mechatronische Systeme

In Bild 5 ist der prinzipielle Aufbau SPS-gesteuerter mechatronischer Systeme dargestellt. Er unterscheidet sich vom Aufbau anderer mechatronischer System lediglich dadurch, dass der Steuerungstyp des Systems schon spezifiziert ist. Bild 3 zeigt zwei Beispiele solcher Systeme wie sie in der Industrie Verwendung finden.

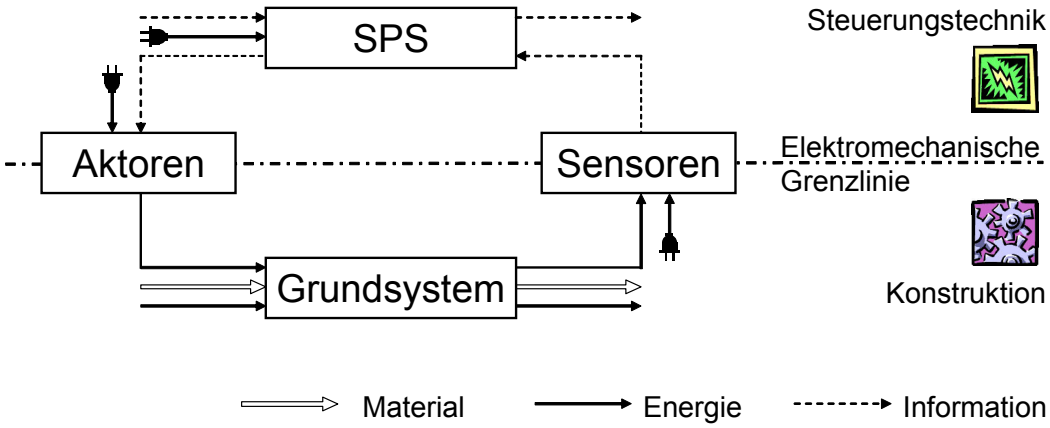


Bild 5: Prinzipieller Aufbau SPS-gesteuerter mechatronischer Systeme inklusive Material-, Energie- und Informationsflüssen

Die SPS sendet Informationen über ihre Ausgänge an die Aktoren. Diese wirken direkt auf das Grundsystem. Sensoren erfassen Zustandsgrössen des Grundsystems und senden Informationen über die Eingänge an die SPS. Das Grundsystem kann Material- und Energieflüsse erhalten und weitergeben. Die SPS ist in der Lage Informationen mit anderen Informationsverarbeitern, wie z.B. einer weiteren SPS oder einem MMI (Mensch-Maschine-Interface) auszutauschen. Traditionell befasst sich der Konstrukteur mit dem elektromechanischen Subsystem, begrenzt durch die elektromechanische Grenzlinie in Bild 5. Also mit allen Teilsystemen, durch die Material und Energie fliesst. Die Steuerungstechnik arbeitet oberhalb dieser Grenzlinie. Für beide Domänen ist die Aktorik und Sensorik relevant. Neben der Auslegung und Auswahl der Aktoren und Sensoren ist aus der Sicht der Konstruktion vor allem die geometrische Ausprägung und Anordnung interessant. Dem gegenüber muss die Steuerungstechnik vor allem Kenntnis über die Belegung der I/O‘s zu der Aktorik und Sensorik besitzen. Bei der Entwicklung SPS-gesteuerter Systeme stellen die Aktoren, die Sensoren und die SPS in der Regel Zukaufteile dar und müssen nicht mitentwickelt werden.

Das Programmieren der SPS wird in Abschnitt 2.3 beschrieben. Die Ausführung des SPS-Programmes wird auf der SPS zyklisch wiederholt (Bild 6).

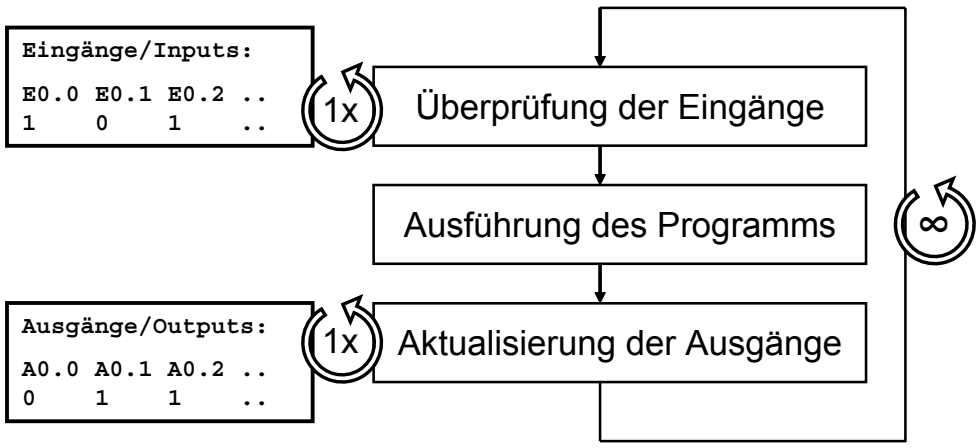


Bild 6: Eingabe-, Verarbeitungs- und Ausgabeteil einer SPS

Eine SPS arbeitet nach dem EVA-Prinzip, sie besitzt also einen Eingabe-, Verarbeitungs- und Ausgabeteil. Die Aktoren und Sensoren sind über die Ausgänge (Outputs) und Eingänge (Inputs) mit der SPS verdrahtet.

Die SPS liest die Werte aller Eingänge am Anfang eines Zykluses ein (man spricht in diesem Zusammenhang auch vom "Einlesen des Prozessabbildes"). Anschliessend führt sie die gespeicherten Programme (auch 'Bausteine' oder 'Netzwerke' genannt) aus und setzt am Ende die Ausgänge. Dann startet der Zyklus von Neuem; ein Programmende gibt es nicht.

Die ersten Speicherprogrammierbaren Steuerungen hatten lediglich digitale Ein- und Ausgänge (digitale I/O‘s), die ausschliesslich die boolschen Werte 0 und 1 akzeptierten. Damit lassen sich z.B. Ventile (de)aktivieren oder Grenzschalter abfragen. Zum Übermitteln kontinuierlicher Werte, wie z.B. der aktuellen Temperatur, sind die so genannten analogen I/O‘s hinzugekommen. Das SPS-Programm spricht alle Ein- und Ausgänge über vordefinierte Variablen an. In der Praxis und in dieser Arbeit hat es sich bewährt einen Präfix im Variablennamen zu verwenden, der die I/O‘s kennzeichnet:

- di, für digitale Inputs
- ai, für analoge Inputs
- do, for digitale Outputs
- ao, for analoge Outputs

So heisst z.B. eine Eingangsvariable einer Lichtschranke zur Detektion von Personen „diPersonDetected“. Diese Input- und Outputvariablen werden parallel zur Programmierung in der SPS-Programmierumgebung inklusive eindeutiger Bitadressierung in der so genannten I/O-Liste angelegt. Beispiele für solche I/O-Listen zu einem SPS-Programm finden sich in Bild 49, in Bild 50 und in Bild 65.

2.2 Konstruktion

Der Produktentwicklungsprozess der Konstruktion ist in Bild 7 der VDI Richtlinie 2221 [73] übersichtlich dargestellt, welche auch für andere Domänen anwendbar ist. Die begriffliche Übertragbarkeit der VDI 2221 auf die Verfahrenstechnik, Feinwerktechnik und Softwareentwicklung ist in [73] nachgewiesen. Das planmässige Vorgehen aus Bild 7 kann darüber hinaus z.B. auch für die Entwicklung von Lehrveranstaltungen [2] eingesetzt werden.

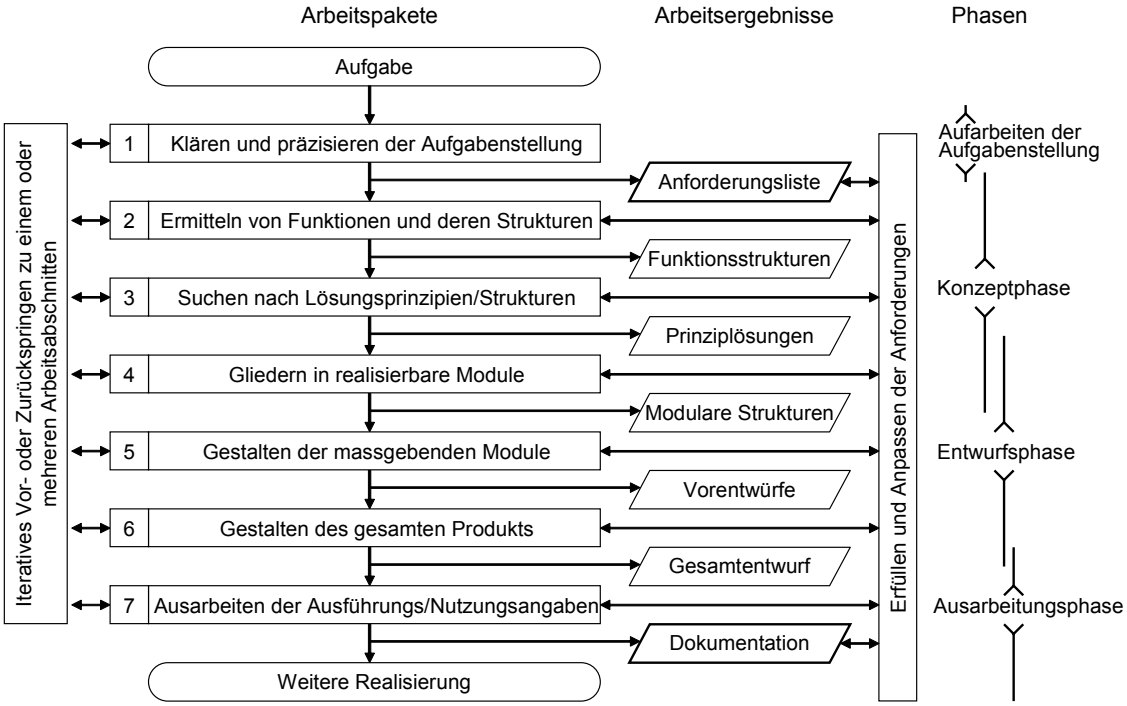


Bild 7: Planmässiges Vorgehen in der Konstruktion nach VDI 2221 [73]

Basierend auf der gegebenen Aufgabe wird im ersten Arbeitsabschnitt die Anforderungsliste erstellt. Durch entwicklungsbegleitende Präzisierung und gegebenenfalls Modifizierung der Aufgabenstellung kann es laufend zu Anpassungen der Anforderungsliste kommen. Dies ist durch

eine Informationsbrücke zwischen Anforderungsliste und Arbeitsabschnitten in Bild 7 gekennzeichnet. Die zweite Brücke (links in Bild 7) stellt sicher, dass die Arbeitsabschnitte nicht starr nacheinander ablaufen müssen. Um schrittweise eine Optimierung zu erreichen, werden die Arbeitsabschnitte häufig iterativ durchlaufen [73].

Im zweiten Arbeitsabschnitt werden Funktionsstrukturen erstellt, um mit Hilfe dieser abstrakten Beschreibung des Systems die Suche nach konkreten Lösungen vorzubereiten. Funktionen beschreiben hier lösungsneutral die Beziehungen zwischen Eingangs-, Ausgangs- und Zustandsgrössen eines Systems [73]. Sie werden durch ein Substantiv und ein Verb im Infinitiv beschrieben, wie z.B. „Flüssigkeit fördern“ [74]. Da die Funktionsstruktur ein wichtiges Element in dieser Arbeit darstellt, wird im Folgenden näher darauf eingegangen.

Die Funktionsstruktur (FS) lässt sich hierarchisch oder ablaufbezogen darstellen [25], [10]. In der hierarchischen Darstellung befindet sich die Gesamtfunktion auf der obersten Hierarchieebene und erfüllt die Gesamtaufgabe des Produktes [73], [10], [59]. Hauptfunktionen beschreiben einen Hauptzweck eines Produktes [73] und dienen unmittelbar der Gesamtfunktion [59]. In der hierarchischen Darstellung der Funktionsstruktur stellen Hauptfunktionen somit Unterfunktionen [25] der Gesamtfunktion dar. Elementarfunktionen sind nach Pahl und Beitz [59] Funktionen, die sich nicht weiter gliedern lassen und allgemein anwendbar sind. Betrachtet man die hierarchische Darstellung der Funktionsstruktur aus der Sicht der Graphentheorie, ergibt sich ein Baum, dessen Wurzel die Gesamtfunktion ist und dessen Blätter Elementarfunktionen sind. In der ablaufbezogenen Darstellung werden Flüsse zwischen den Funktionen eingetragen. Dabei wird zwischen Material-, Energie- und Informationsflüssen unterschieden [10], [59]. In Bild 8 sind beide Darstellungen gegenübergestellt.

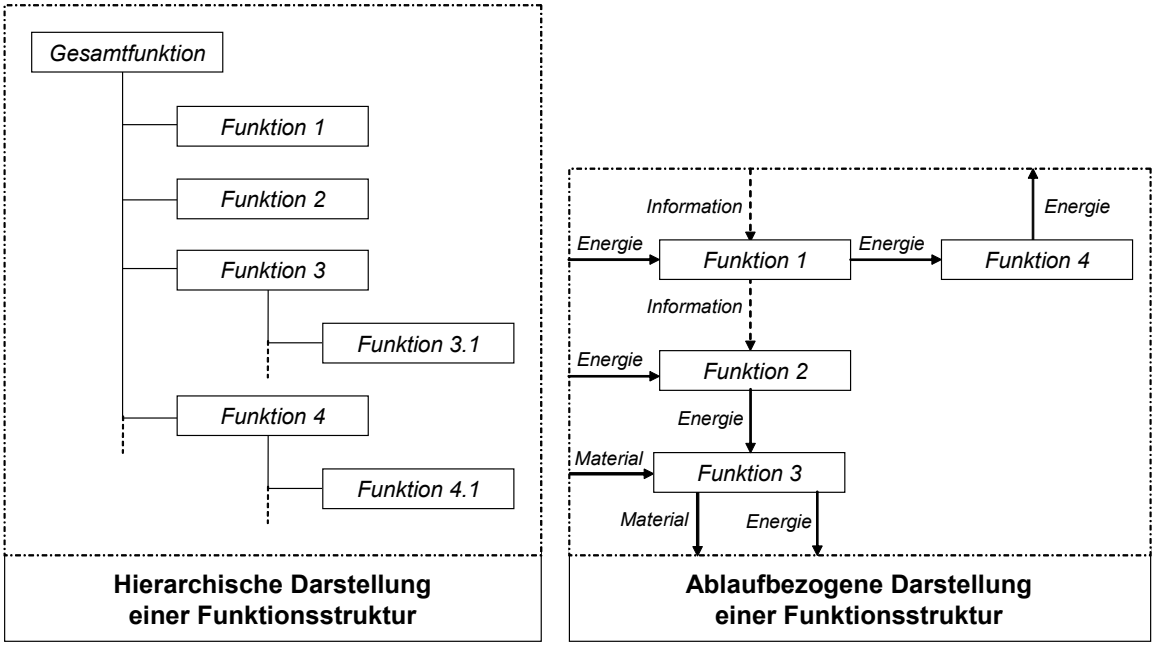


Bild 8: Hierarchische und ablaufbezogene Darstellung einer Funktionsstruktur

In diesem Beispiel ist die Funktion 1 gleichzeitig Hauptfunktion, Unterfunktion und Elementarfunktion. Die Hauptfunktionen 1 bis 4 auf der ersten Hierarchiestufe sind rechts in Bild 8 exemplarisch ablaufbezogen dargestellt. Bewährt hat sich in dieser Arbeit die Formulierung von pragmatischen Funktionen nach Eisenhut [25]:

„Die (pragmatische) Funktion beschreibt den Zweck (die Aufgabe) eines Produktes oder eines Teils eines Produktes. Sie wandelt eine gegebene Eingangs- in eine gewünschte Ausgangsgröße um, indem sie Eigenschaften verändert. Die Pragmatische Funktion wird immer lösungsneutral formuliert und beinhaltet mindestens ein Verb und ein Objekt, auf das sich das Verb bezieht, Die Formulierung kann durch Subjekt, Adjektiv oder Adverb präzisiert werden.“

Der Abstraktionsgrad bei den pragmatischen Funktionen ist nicht so gross wie bei der Formulierung der traditionellen Funktionen aus der Konstruktionsmethodik. Dadurch sollen Lösungen ausgeschlossen werden, die im aktuellen Kontext keinen Sinn machen. Der wesentliche Vorteil besteht darin, dass es einfacher wird diese Funktionen zu beschreiben und zu interpretieren. Dies wird anhand der Gegenüberstellung einiger Funktionen einer Druckmaschine offensichtlich (Bild 9)

.

Pragmatische Funktion	Konstruktionsmeth. Funktion
Bogen anlegen	Stoff leiten
Stapel im Anleger handhaben	Stoff leiten
Stapel in Anleger einführen	Stoff leiten
…	
Hilfsstapel im Anleger bilden	Stoff verzweigen
Haupt- mit Hilfsstapel im Anleger vereinen	Stoff verknüpfen
…	

Bild 9: Pragmatische Funktionen im Vergleich zu konstruktionsmethodischen Funktionen für einen Anleger einer Druckmaschine aus [25]

Obwohl die pragmatischen Funktionen konkreter sind als die Konstruktionsmethodischen, werden die einzelnen Teillösungen für den Anleger der Druckmaschine noch nicht vorweggenommen. Im Arbeitsabschnitt 3 der VDI Richtlinie 2221 (Bild 7) werden Prinziplösungen erarbeitet. Zum Sammeln und Kombinieren der Prinziplösungen eignet sich der Morphologische Kasten wie er in Bild 10 dargestellt ist.

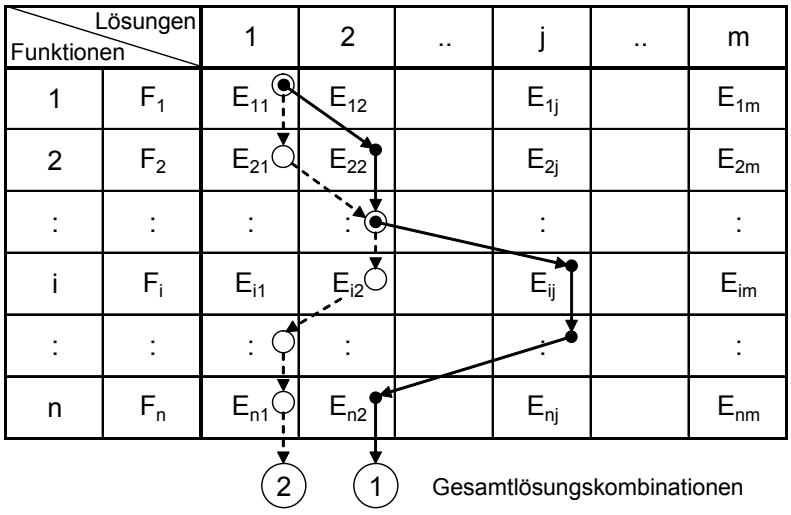


Bild 10: Kombination von Einzellösungen zu Gesamtlösungen im Morphologischen Kasten nach Pahl und Beitz [59]

Pro Zeile ist jeweils eine Funktion (F) aus dem vorhergehenden Arbeitsabschnitt 2 einzutragen. Dann werden verschiedene Einzellösungen (E) zur Umsetzung der einzelnen Funktionen diskutiert, kombiniert und bewertet.

Das Bilden von Modulen in Arbeitsabschnitt 4 vor den arbeitsintensiven Gestaltungsschritten ist insbesondere bei komplexen Produkten wichtig, um eine effiziente Aufteilung der Konstruktionsarbeit zu erleichtern (concurrent engineering innerhalb der Konstruktion) und durch Strukturierung bestimmte Entwicklungsschwerpunkte besser erkennen zu können [73]. In Abschnitt 2.4.2 wird eine Methode vorgestellt, die eine Modularisierung aufgrund der ablaufbezogenen Darstellung einer Funktionsstruktur ermöglicht.

Sind die modularen Strukturen in Arbeitsabschnitt 4 der VDI Richtlinie 2221 (Bild 7) erstellt, können die Module auf mehrere Konstrukteure verteilt werden, um die Entwürfe in den nachfolgenden Arbeitsabschnitten 5-7 zu erarbeiten und zu dokumentieren. Dazu wird in der Regel ein CAD-System verwendet.

FOD (Function Oriented Design) [15] ist eine Software, die die Konzeptphase basierend auf der VDI 2221 (Bild 7) unterstützt und eine CAD-Ankopplung anbietet. In Bild 11 ist die Modellstruktur der Software FOD illustriert. Zur Erfassung der Anforderungs-, Funktions-, Struktur- und Constraintmodelle existsiert jeweils ein eigener Editor. Zur gezielten Wiederverwendung von Teilmodellen können diese über den Bibliotheksmanager als Komponente abgelegt werden.

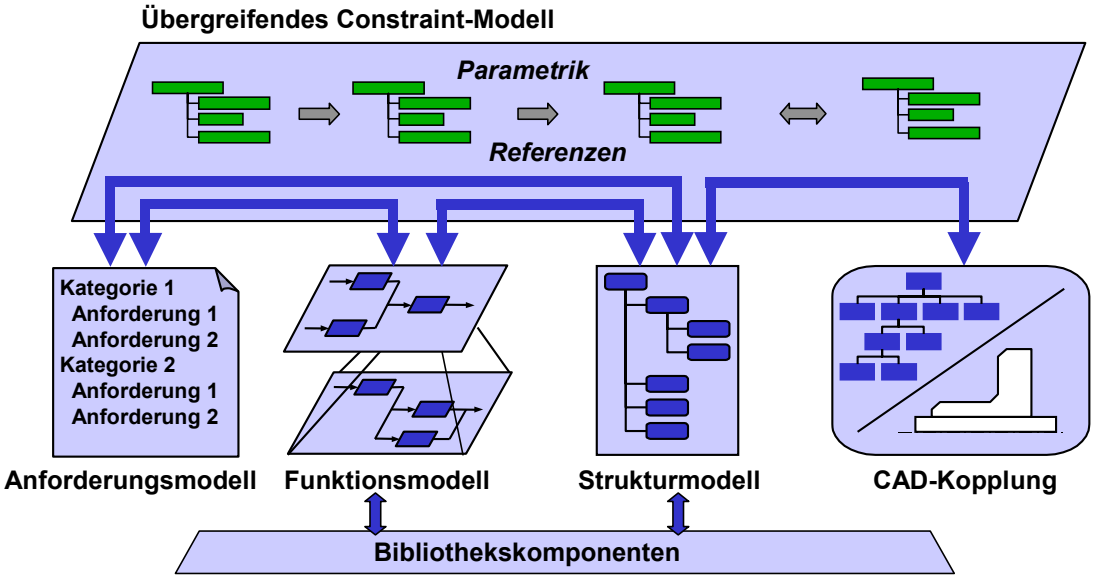


Bild 11: Modellstruktur der Software FOD [15]

Im Anforderungseditor ist es möglich, neben einem digitalen Pflichtenheft alle Projektdokumente strukturiert im Anforderungsmodell abzulegen. FOD unterstützt im Funktionseditor die hierarchische und ablaufbezogene Darstellung der Funktionsstruktur. Der Bauteileditor dient zur Erfassung der Baugruppenhierarchie, welche im Strukturmodell abgelegt wird. Zusätzlich lassen sich über den Constraintmanager Parameter der Anforderungs-, Funktions- und Strukturmodelle miteinander verknüpfen. Darüber hinaus stehen Schnittstellenmodule für verschiedene CAD-Systeme zur Verfügung. Dadurch können Baugruppen im CAD bidirektional mit den Baugruppen im FOD verbunden werden. Somit eignet sich FOD zur durchgängigen Unterstützung des Entwicklungsprozesses in der Konstruktion nach VDI 2221 (Bild 7). Allerdings fehlt die direkte Unterstützung zur Unterscheidung der Material-, Energie- und Informationsflüsse im Funktionseditor. Für den Entwicklungsprozess der Steuerungstechnik welcher im nächsten Abschnitt beschrieben wird ist FOD nicht ausgelegt.

2.3 Steuerungstechnik

Wie in Abschnitt 1.1 dargestellt, zieht die Konstruktion die Steuerungstechnik in der Regel nicht vor Abschluss der Konzeptphase (Bild 7) hinzu. Oft steigt die Steuerungsstechnik erst nach der Entwurfsphase ein. Zu Beginn werden oft verschiedene Diagramme erstellt, um die Aufgabenstellung aus Sicht der Steuerungstechnik zu erfassen. Dadurch erhält der Steuerungstechniker einen Einblick in die grundsätzlichen Abläufe und Abhängigkeiten zwischen den Aktoren und Sensoren. Dies geschieht entweder direkt auf Papier oder in allgemeinen 2D-Graphiktools wie Visio [52]. Zum Teil kommen auch Speziallösungen zum Einsatz ([26], [54]).

Neben nicht standardisierten Flussdiagrammen finden dazu Funktionsdiagramme ([75], [19], [80]) und Funktionspläne ([22], [18], [77], [80]) Anwendung. In Bild 12 sind beide Diagrammtypen der Praxis gegenübergestellt. Der Funktionsplan (Bild 12) zur Beschreibung der Steuerungsaufgabe darf nicht mit der SPS-Programmiersprache FUP (Funktionsplan) verwechselt werden.

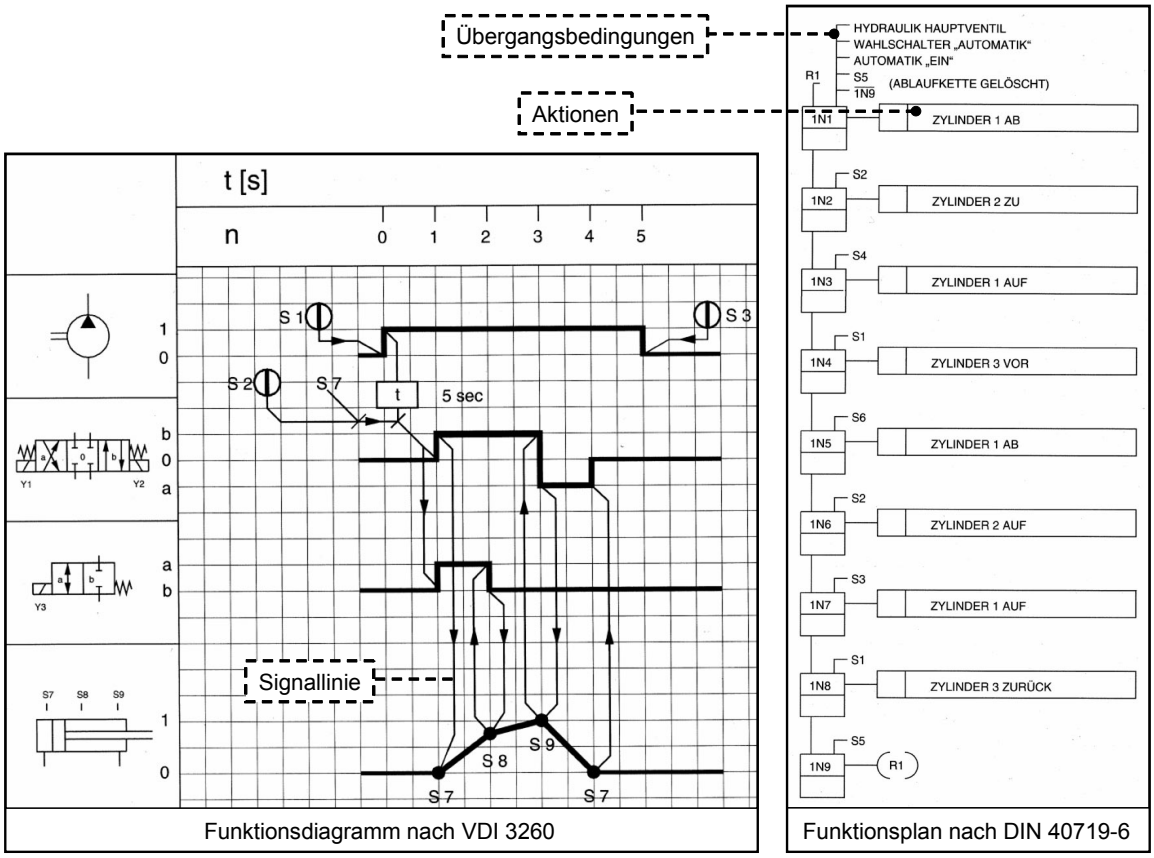


Bild 12: Funktionsdiagramm nach VDI 3260 und Funktionsplan nach DIN 40719 aus [80]

In der Vertikalen können im Funktionsdiagramm die Wege der Aktoren eingetragen werden; in der Horizontalen die Schritte. Über so genannte Signallinien ist der Signalfluss der beteiligten Sensoren repräsentiert. Solche Funktionsdiagramme nach VDI 3260 [75] werden oft auch Weg-Schrittdiagramm genannt. Ist anstelle der Schritte das zeitliche Verhalten eingetragen, spricht man auch von einem Weg-Zeitdiagramm. Obwohl die VDI 3260 ersatzlos zurückgezogen wurde, sind in ihr Funktionsdiagramme genormt wie sie auch heute noch zum Einsatz kommen. So bietet z.B. der Funktionsdiagrammeditor der Software Fluiddraw [26] von der Firma FESTO die Möglichkeit solche Diagramme zu erstellen. Insbesondere für kleinere (Teil-)Systeme kann das Zusammenspiel einiger Aktoren und Sensoren über Funktionsdiagramme gut veranschaulicht werden.

Anstelle der VDI 3260 empfiehlt der VDI die Anwendung von DIN 40719-6 [18], da sich umfangreiche Steuerungen mit Hilfe der Funktionsdiagramme nicht mehr in allen Details übersichtlich darstellen lassen [80]. Der Funktionsplan nach DIN 40719-6 (Bild 12) ist

ablauforientiert und besteht hauptsächlich aus Übergangsbedingungen und Aktionen. Übergangsbedingungen überprüfen den Status definierter Sensoren. Falls die Übergangsbedingung erfüllt ist, wird eine Aktion über einen oder mehrere Aktoren ausgeführt. Dadurch wird der Ablauf Schritt für Schritt dokumentiert. Grössere Systeme lassen sich über hierarchische Verschachtelung modellieren. Die DIN 40719-6 ist mittlerweile durch die aktuelle Norm DIN EN 60848 [22] ersetzt worden. In dieser Norm wird der Funktionsplan GRAFCET (GRAphe Fonctionnel de Commande Etape Transition) standardisiert. GRAFCET ähnelt im Erscheinungsbild dem Funktionsplan nach DIN 40719, ist aber klarer spezifiziert und orientiert sich an der SPS-Programmiersprache SFC (Sequential Function Chart) aus der DIN EN 61131-3 [23]. Sind die Abläufe über GRAFCET konzipiert, können sie einfach als SFC-Programm implementiert werden. Analog zur grafischen Entwurfssprache GRAFCET für die funktionale Beschreibung des Verhaltens einer SPS, implementiert SFC die Abläufe schrittweise über Aktionen und Transitionen (Bild 13)..

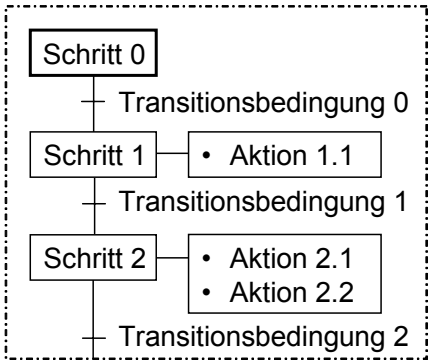


Bild 13: SFC-Struktur nach DIN EN 61131-3 [23]

Der Status der Sensoren wird in den Transitionen abgefragt und logisch verknüpft. In den Aktionen werden Befehle an die Aktoren spezifiziert. Zusätzlich besteht die Möglichkeit, innerhalb einer Aktion ein weiteres Unterprogramm in SFC oder einer anderen Sprache aufzurufen. Neben SFC sind dazu vier weitere SPS-Programmiersprachen in der DIN EN 61131-3 normiert:

- SFC, Sequential Function Chart (AS, Ablaufsprache)
- ST, Structured Text (ST, Strukturierter Text)
- FBD, Function Block Diagram (FUP, Funktionsplan)
- IL, Instruction List (AWL, Anweisungsliste)
- LD, Ladder Diagram (KOP, Kontaktplan)

Die Hochsprache ST ähnelt der Programmiersprache PASCAL. Sie stellt Operatoren (wie z.B. SIN, OR, MOD, ..) und Anweisungen (wie z.B. IF, CASE, FOR, WHILE, ..) für die prozedurale Programmierung zur Verfügung. FBD heisst zwar auch Funktionsplan (FUP) in der deutschen Übersetzung, darf aber nicht mit den Funktionsplänen der Normen DIN 40719-6 [19] oder DIN EN 60848 [22] verwechselt werden. FBD erlaubt das boolsche Verschalten von Logikgattern (Bild 47). IL setzt sich aus einer Folge von Anweisungen zusammen. Jede Anweisung muss einen Operator (z.B. AND) mit optionalen Modifizierern enthalten. Der Modifizierer N zeigt z.B. die boolesche Negation des Operanden (auf den der Operator wirkt) an. So ist die Anweisung „ANDN B“ als
„aktueller Wert := aktueller Wert AND NOT B“
zu verstehen. LD kommt den Programmierern entgegen, die schon Steuerungen in Relaistechnik entwickelt haben. Ein bestehender Relaisstromlaufplan kann via LD meist direkt für eine SPS implementiert werden [79]. Aktoren werden dabei über „Relais“, Sensoren über „Kontakte“ angesprochen.

Zur Implementierung dieser Sprachen kommen SPS-Programmierumgebungen wie z.B. SIMATIC von SIEMENS [65] oder das Automation Studio von B&R [7] zum Einsatz. Dabei sind alle fünf genormten Programmiersprachen miteinander in Kombination verwendbar.

Bild 14 zeigt die empfohlene Verteilung der SPS-Programmiersprachen auf die Phasen des Softwareentwicklungsprozesses. Zusätzlich ist in dem Bild von Bonfatti, Monari und Sampieri [8] der Abstraktionsgrad der Sprachen illustriert.

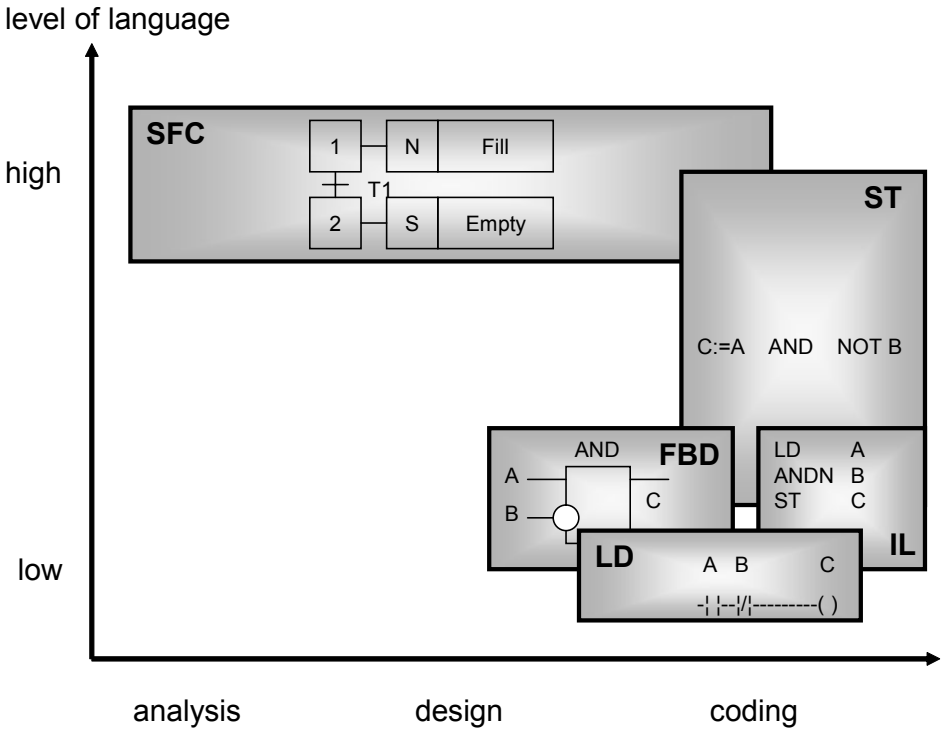


Bild 14: Verteilung der fünf normierten SPS-Programmiersprachen (DIN EN 61131-3, [23]) auf die verschiedenen Phasen im Entwicklungsprozess für SPS-Software in Anlehnung an [8]

Die Nähe zu GRAFCET prädestiniert SFC als Programmiersprache für den Einstieg in die Softwareentwicklung. Werden Abläufe in SFC dargestellt sind sie zudem auch für Mitarbeiter ausserhalb der Steuerungstechnik einfacher nachvollziehbar. Für das weitere Ausprogrammieren stellt das Beiblatt 1 zur DIN EN 61131-3 [24] Leitlinien für die Anwendung und Implementierung zur Verfügung. Durch die DIN EN 61131-3 [23] wird der Top-down Entwurf unterstützt, bei dem ein Entwurf durch funktionelle Zerlegung (mittels SFC) stattfindet. Dieser funktionale Zerlegungsprozess wird so lange durchgeführt, bis jede Funktionalität entweder als ein Element in einer vorhandenen Bibliothek gefunden werden kann oder bis sie in einer der textuellen oder grafischen Sprachen algorithmisch ausgedrückt werden kann. Das System wird dann „bottom-up“ mittels funktionaler Komposition implementiert; d. h. durch Zusammensetzen und Hinzufügen der neu definierten Elemente zur Bibliothek in der umgekehrten Reihenfolge, in der sie in den vorangegangenen Schritten definiert wurden. Wenn während des Entwurfs darauf geachtet wurde, können viele der neuen Bibliothekselemente in künftigen Systementwicklungen wieder verwendet werden. Dies trägt zur Zuverlässigkeit, Instandhalt-

barkeit, Benutzbarkeit und Anpassbarkeit der Software bei [24]. Zum Schluss werden durch Anwenden der Konfigurationselemente die Zuordnung der Programme zu den Ressourcen und der Ressourcen zu den Konfigurationen vervollständigt, die Tasks für die Programmausführung festgelegt, die Zugriffspfade für die Kommunikation mit den Informationssystemen eingerichtet und das SPS-Programm auf die SPS geladen.

2.4 Interdisziplinäre Entwicklung

Im Fokus dieser Arbeit steht die interdisziplinäre Entwicklung SPS-gesteuerter mechatronischer Systeme. Sind mehrere Domänen bei der Entwicklung eines Produktes beteiligt, reicht die Strukturierung des Vorgehens nach VDI 2221 [73] nicht mehr aus. Im Sinne einer Partitionierung weist die VDI 2422 [76] die Funktionserfüllung den einzelnen Domänen zu. Eine detaillierte Anweisung, wie eine integrierte Wirkstruktur erzeugt werden kann, erfolgt aber nicht [44]. Die aktuelle VDI Richtlinie 2206 [72] soll die Grundzüge des Entwickelns mechatronischer Systeme vermitteln und zu einer ganzheitlichen Sichtweise über die einzelne Fachdisziplin hinaus anregen. Sie positioniert sich ergänzend zu den Richtlinien VDI 2221 und VDI 2422. Analog der Richtlinie VDI 2221 werden in der Richtlinie VDI 2206 die Methoden zur Entwicklung mechatronischer Systeme beschrieben. Die mechatronischen Ansätze der Richtlinie VDI 2422, die vor dem Hintergrund des Einzugs der Mikroelektronik in die Gerätetechnik entstand, werden erweitert und zu einem durchgängigen domänenübergreifenden Leitfaden ausgebaut [72].

In der VDI 2206 wird ein flexibles Vorgehensmodell vorgeschlagen, das sich im Wesentlichen auf drei Elemente stützt:

- allgemeiner Problemlösungszyklus auf der Mikroebene
- V-Modell auf der Makroebene
- vordefinierte Prozessbausteine zur Bearbeitung wiederkehrender Arbeitsschritte bei der Entwicklung mechatronischer Systeme

Der Problemlösungszyklus auf der Mikroebene stammt aus dem Systems Engineering [30]. Durch Aneinanderreihen und Verschachteln von Vorgehenszyklen lässt sich die Prozessplanung flexibel an die Eigenheiten jeder Entwicklungsaufgabe anpassen. Der Mikrozyklus der

VDI 2206 soll vor allem dem im Prozess stehenden Produktentwickler bei der Bearbeitung vorhersehbarer und damit planbarer Teilaufgaben, aber auch bei der Lösung plötzlich auftretender, unvorhersehbarer Probleme unterstützen [72].

Das V-Modell auf der Makroebene ist in Bild 15 dargestellt. Es beschreibt das generische Vorgehen beim Entwurf mechatronischer Systeme, das fallweise auszuprägen ist. Im Gegensatz zur VDI 2221 [73] schliesst hier der Ausdruck Entwurf die Konzeptphase aus Bild 7 mit ein.

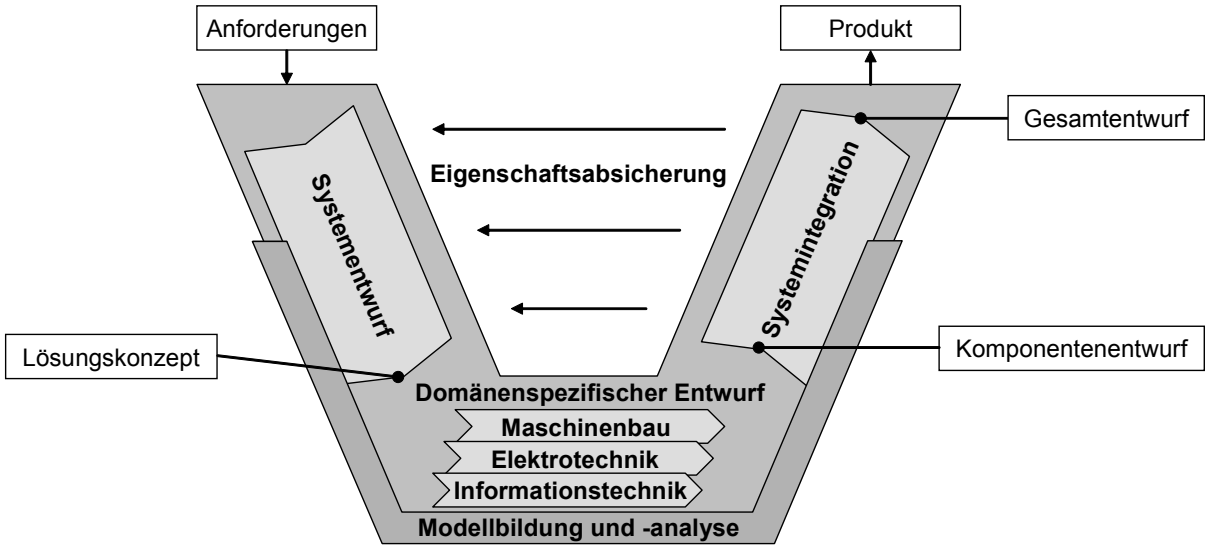


Bild 15: Das V-Modell der VDI 2206 [72]

Basierend auf den Anforderungen wird domänenübergreifend im Systementwurf das Lösungskonzept erarbeitet. Im domänenspezifischen Entwurf findet eine Partitionierung der Aufgaben statt und jede Domäne arbeitet ihren Entwurf aus. Bei der Entwicklung SPS-gesteuerter mechatronischer Systeme wird dazu hauptsächlich das CAD in der Konstruktion und die SPS-Programmierumgebung in der Steuerungstechnik verwendet. Während der Systemintegration wird der Komponentenentwurf der einzelnen Domänen zu einem Gesamtentwurf integriert, um das Zusammenwirken untersuchen zu können. Das Produkt ist das Ergebnis eines durchlaufenen Makrozyklus. Wobei unter dem Wort Produkt nicht ausschliesslich das real existierende Erzeugnis für den Kunden verstanden wird, sondern die zunehmende Konkretisierung des zukünftigen Produktes (Produktreife). Reifegrade sind z.B. das Labormuster, das Funktionsmuster und das Vorserienprodukt [72]. So führt das mehrmalige Durchlaufen des V-Modells der

VDI 2206 zu einer zunehmenden Produktreife. In der Modellbildung und -analyse (Bild 15) wird der Entwurf flankiert. Die Systemeigenschaften werden mit Hilfe von Modellen und rechnerunterstützten Werkzeugen zur Simulation abgebildet und untersucht. Ein Beispiel für solch eine Modellbildung findet sich in [38]. Während des Entwurfs muss fortlaufend eine Eigenschaftsabsicherung (Bild 15) anhand des spezifizierten Lösungskonzepts und der Anforderungen stattfinden. Eine Überprüfung der gewünschten Systemeigenschaften von SPS-gesteuerten mechatronischen Systemen lässt sich z.B. mit der Virtuellen Maschine [45], [17] realisieren (Bild 16).

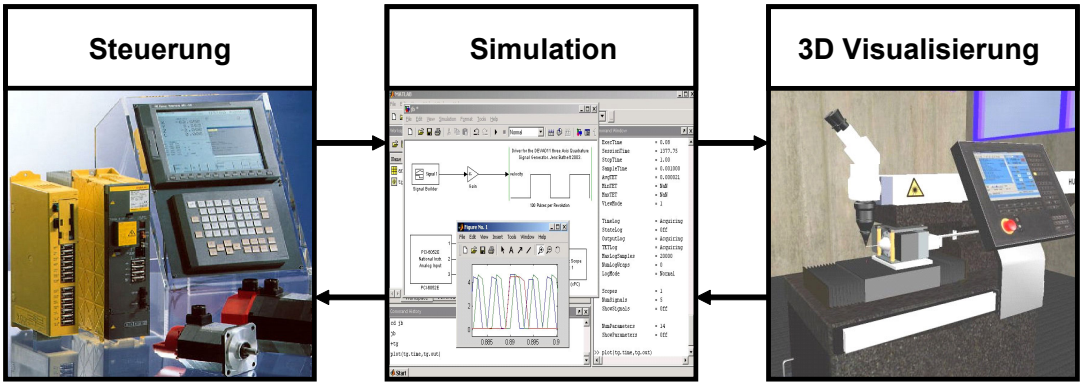


Bild 16: Das Konzept der Virtuellen Maschine aus [5]

In der Virtuellen Maschine ist die Maschinensteuerung so mit einer Maschinensimulation verknüpft, dass die Simulation Aktorsignale empfängt, verarbeitet und virtuelle Sensorsignale in Echtzeit wieder an die Steuerung zurücksendet. Zusätzlich sind Geometriedaten aus dem CAD angehängt, so dass Bewegungen visualisiert und Events wie „Notknopf gedrückt“ oder „Werkzeug/Werkstück-Kollision“ direkt an die Simulation oder weiter an die Steuerung übermittelt werden können. Je nach Fragestellung können dabei in der Eventsimulation die Events einfach verknüpft und das Verhalten der Maschine grob linearisiert werden [17]. Zur weitergehenden Betrachtung des Systemverhaltens können dazu aber auch Modelle aus der flankierenden Modellbildung und -analyse integriert werden [40], [5].

Neben dem allgemeinen Problemlösungszyklus auf der Mikroebene und dem V-Modell auf der Makroebene stützt sich die VDI 2206 auf vordefinierte Prozessbausteine zur Bearbeitung wie-

derkehrender Arbeitsschritte bei der Entwicklung mechatronischer Systeme. Der vordefinierte Prozessbaustein für den Systementwurf ist in Bild 17 gezeigt.

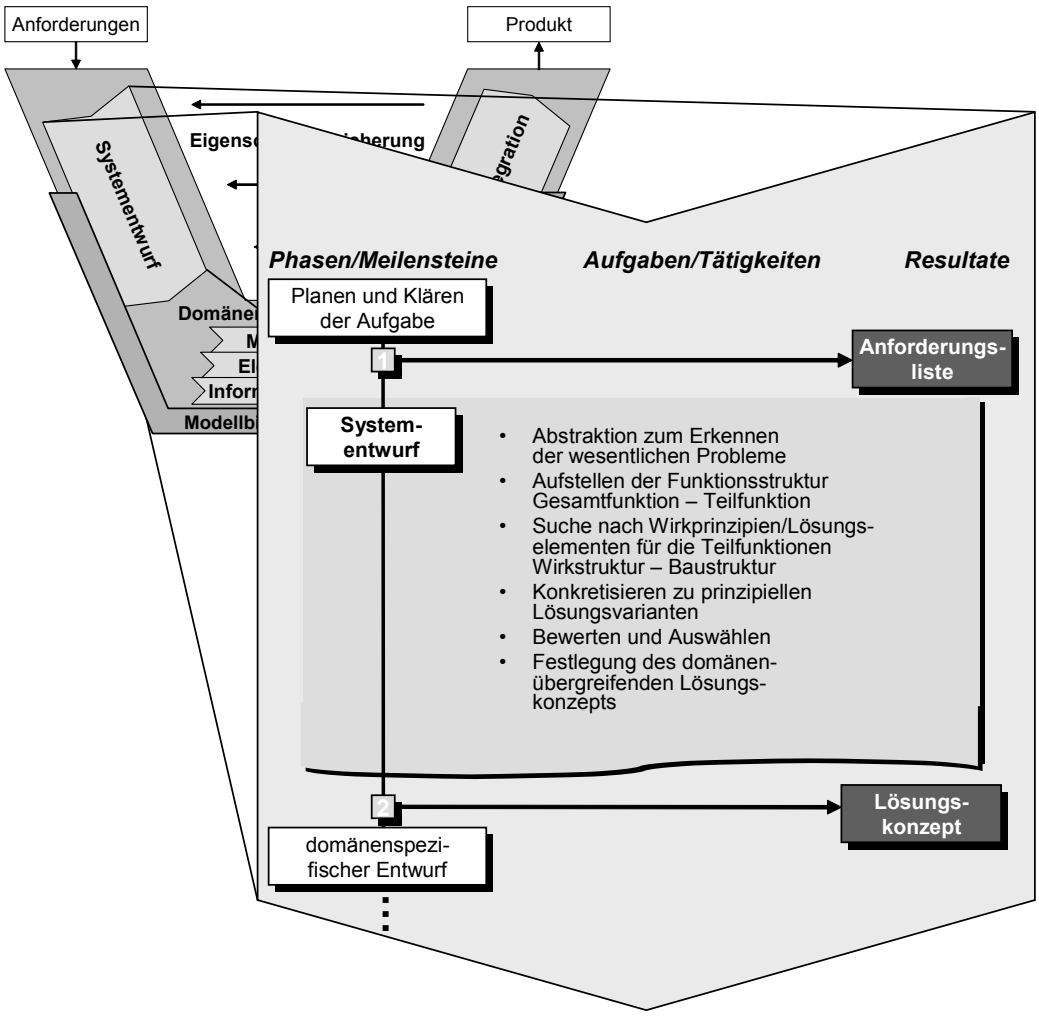


Bild 17: Vordefinierter Prozessbaustein für den Systementwurf der VDI 2206 [72]

Die aufgeführten Aufgaben und Tätigkeiten lehnen sich an das Vorgehen von Pahl und Beitz [59] an. Parallelen zur Konzeptphase (Bild 7) nach VDI 2221 [73] sind offensichtlich. In der VDI 2206 existieren weitere Prozessbausteine zur konkreteren Beschreibung der Modellbildung und -analyse, des domänenspezifischen Entwurfs, der Systemintegration und der Eigenschaftsabsicherung.

In den folgenden Abschnitten werden weitere Ansätze im Kontext dieser Arbeit beschrieben.

2.4.1 Buur

Buur fordert in [12] eine interdisziplinär akzeptierte, les- und schreibbare Konzeptbeschreibungssprache zur lösungsneutralen und funktionalen Beschreibung mechatronischer Systeme. In [13] konkretisiert Buur die Beschreibung einer mechatronischen Funktion durch die folgenden Komponenten:

- Zustände (states) und Übergangsbedingungen (transitions), die die Logik abbilden welche Transformationsfunktionen wann ausgeführt wird.
- Transformationsfunktionen (transformation functions), die wie in [59] beschrieben, Material, Energie und Informationen transformieren.
- Zweckfunktionen (purpose functions), die die notwendigen Effekte zur Verfügung stellen um die Transformationen auszuführen.
- Aktive Zweckfunktionen (active purpose functions), die für jeden möglichen Zustand des Systems aus den Zweckfunktionen ausgewählt werden müssen.

Am Beispiel eines Telefons zeigt Buur in Bild 18 diese funktionelle Darstellung.

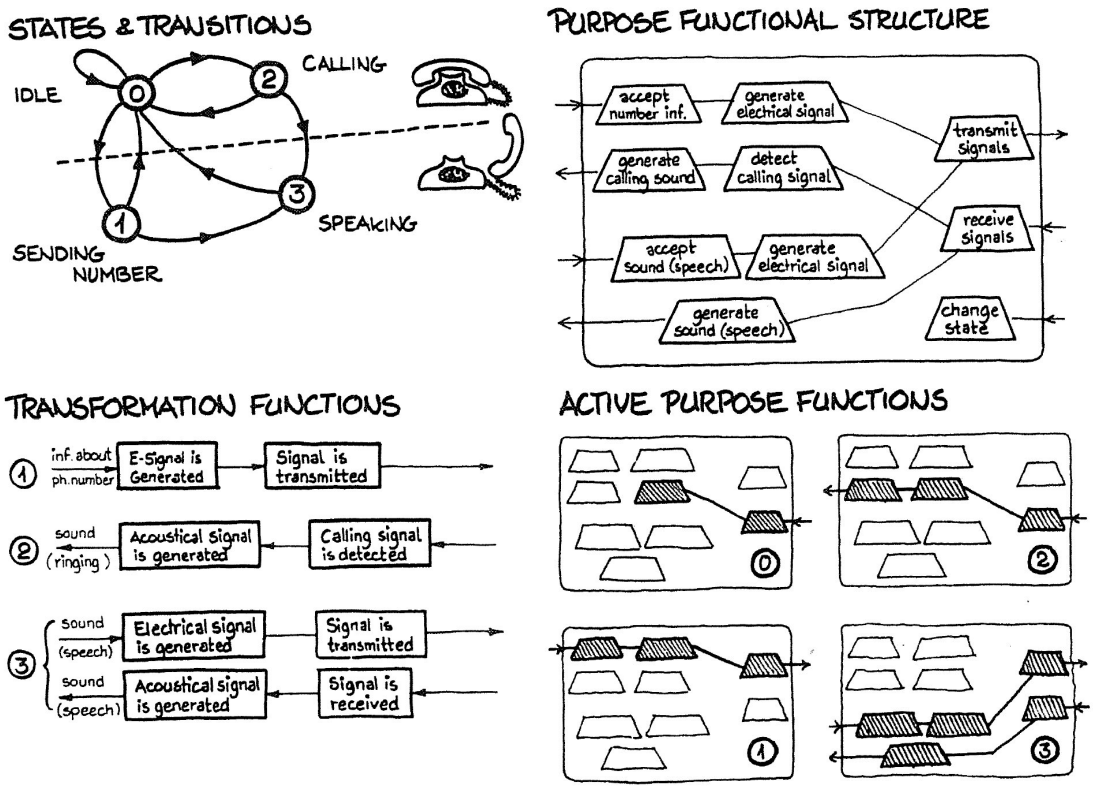


Bild 18: Funktionen eines Telefons nach Buur [13]

Für die Zustände „1 - sending number“, „2 - calling“ und „3 - speaking“ sind die Transformationsfunktionen dargestellt. Zusätzlich sind die Zweckfunktionen und die je nach Zustand (0-3) aktiven Zweckfunktionen aufgeführt.

Später hat Buur das ‚Mechatronic Design Concept' in [14] präsentiert. Es stellt eine Prinziplösung dar, die durch vier Eigenschaften beschrieben wird:

1. Zustände und Übergänge
2. Abläufe
3. Hierarchie
4. Timing

Buur hat somit wichtige Kriterien für die Beschreibung von mechatronischen Konzepten formuliert.

2.4.2 Stone

Stone schlägt in [67] eine Modularisierung basierend auf der ablaufbezogenen Darstellung einer Funktionsstruktur (Bild 8) vor. Dazu hat er die drei heuristischen Regeln „Dominant flow“, „Branching flow“ und „Conversion-transmission“ entwickelt, um potentielle Module aus einer Funktionsstruktur abzuleiten. Die Heuristik „Dominant flow“ ist in Bild 19 illustriert.

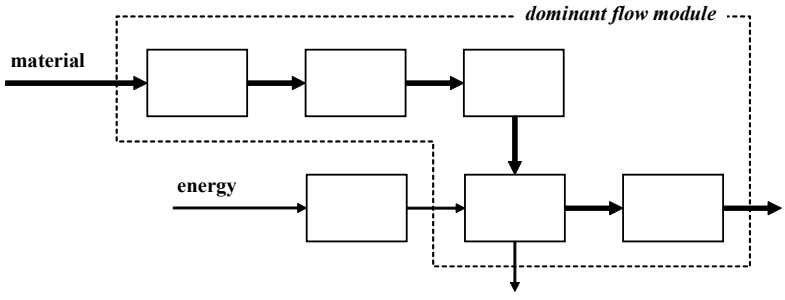


Bild 19: Regel „Dominant flow“ nach Stone [67]

Hier wird jeder Fluss der Funktionsstruktur verfolgt, bis der entsprechende Fluss entweder das System verlässt oder aber in einen anderen Flusstyp umgewandelt wird. Die Funktionen, durch

die der untersuchte Fluss verläuft, werden dann anschliessend in potentielle Module zusammengefasst. Dabei hat Stone diese und die beiden folgenden Heuristiken auf elektromechanische Systeme mit Material und Energieflüssen angewandt.

In der zweiten Heuristik „Branching flow“ (Bild 20) werden Flüsse identifiziert, die sich in parallele Funktionsflussketten verzweigen.

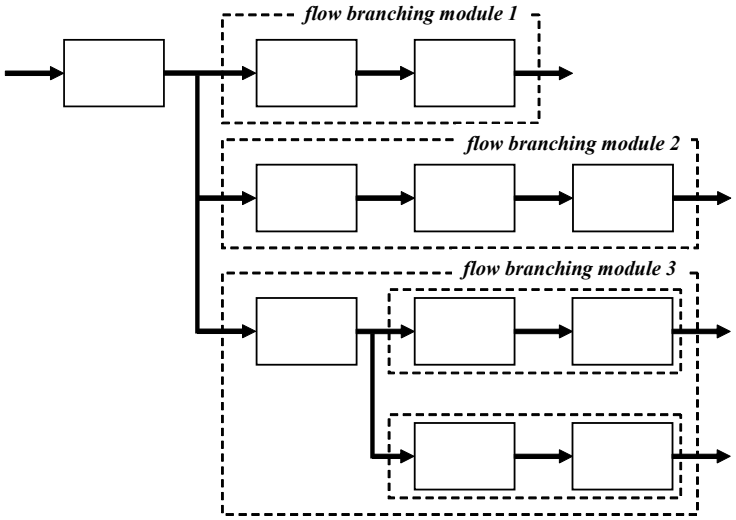


Bild 20: Regel „Branching flow“ nach Stone [67]

Die Funktionen jedes einzelnen Astes dieser Flussverzeigung werden wiederum gruppiert und stellen so weitere mögliche Module dar.

Die Heuristik „Conversion-transmission“ (Bild 21) beschäftigt sich mit Funktionen, die die Eingangsflüsse in Ausgangsflüsse einer anderen Form wandeln, sowie mit Funktionsflussketten, die diese gewandelten Flüsse weiterleiten.

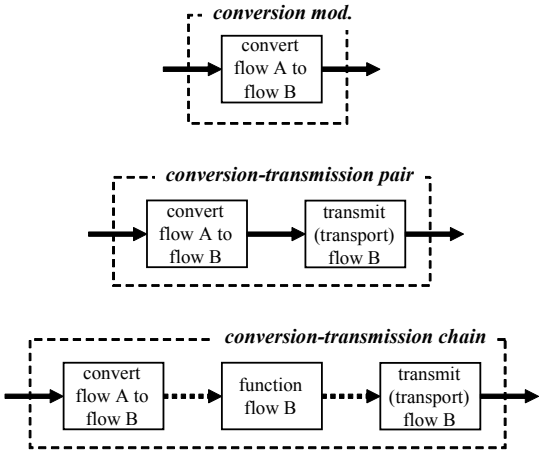


Bild 21: Regel „Conversion-transmission“ nach Stone [67]

Die Wandelfunktionen können eigenständige Module bilden. Wenn den Wandelfunktionen Funktionen folgen, die die gewandelten Flüsse weiterleiten, ohne dass sie diese wandeln, so werden diese Funktionen ebenfalls im entsprechenden Modul integriert.

Die Anwendung der drei Heuristiken auf die Funktionsstruktur des Systems führt zu verschiedenen Vorschlägen zur Modularisierung. Abschliessend werden die potentiellen Module anhand der Anforderungen bewertet und die entgültigen Module ausgewählt [67].

Diese Methode erlaubt somit das strukturieren elektromechanischer Subsysteme. Die ausgewählten Module können als Baugruppen im CAD ausmodelliert werden. Der Softwareentwurf bleibt unbeeinflusst.

2.4.3 Kallenbach

Das in Bild 22 dargestellte Phasenmodell von Kallenbach [41] zur Entwicklung mechatronischer Systeme basiert auf dem Vorgehen der VDI Richtlinie 2221 [73].

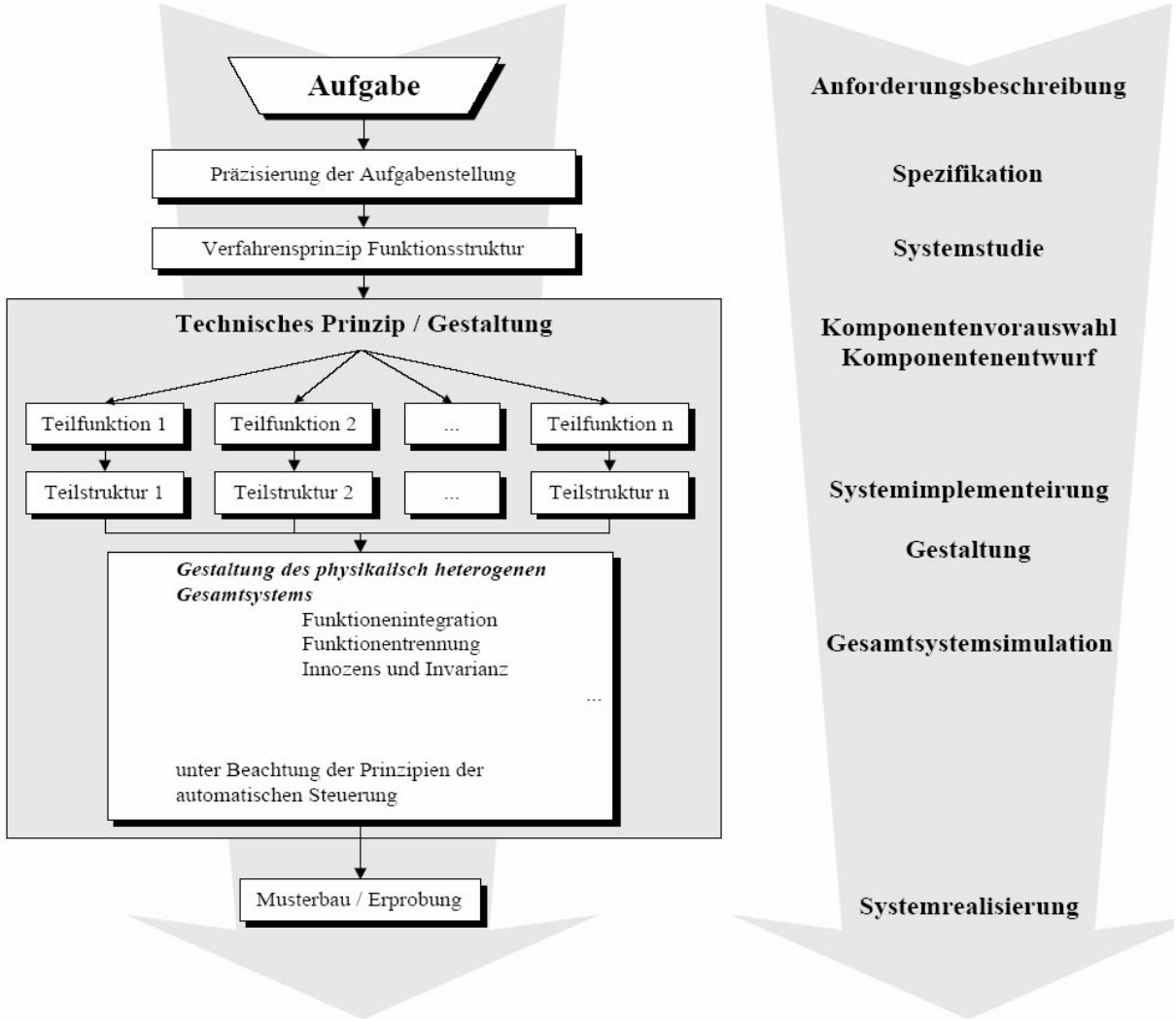


Bild 22: Phasenmodell mechatronischer Systeme nach Kallenbach [41] (für Stellantriebe ist in [42] ein analoges Phasenmodell dargestellt)

Analog zur VDI 2221 (Bild 7) wird im ersten Arbeitsabschnitt die Aufgabenstellung präzisiert. Anschliessend wird eine hierarchische Funktionsstruktur inklusive Störfunktionen erstellt. Die Gestaltung des physikalisch heterogenen Gesamtsystems umfasst die Zerlegung der Gesamtfunktion in Teilfunktionen und das Gestalten der Elemente, die diese Teilfunktionen möglichst optimal erfüllen. Zur Verkürzung der Entwicklungszeit fordert Kallenbach eine Sammlung bewährter Funktionsstrukuren (Funktionsstrukturspeicher) und wissensbasierte Entscheidungshilfen zur Auswahl der Funktionsstrukturen auf unterschiedlichen Hierarchieebenen. Eine Analyse der Funktions-Gestalt-Beziehungen über geeignete Entwurfswerkzeuge optimiert die gesamte Gestalt des Systems. Durch das Gliedern in realisierbare Subkomponenten ergibt sich eine Aufteilung in die Domänen Mechanik, Elektrik und Softwaretechnik. Pro Komponente

folgt eine Phase der Strukturfestlegung, der Prinzipauswahl und der Variantenbildung mit anschliessender Bewertung und Auswahl. Kallenbach fordert die Entwicklung offener Entwurfssysteme, die durch gut strukturierte Datenspeicher den Entwurf effektiv unterstützen, ohne den Entwickler bei der Lösungssuche zu stark einzuengen. Gleichzeitig muß ein derart gestaltetes Entwufssystem durch domänenspezifische Entwurfssoftware erweiterbar sein.

2.4.4 Flath

Flath stellt in [27] eine integrierte Methode zur Funktions- und Prinziplösungsmodellierung mechatronischer Produkte vor, welche eine Spezifikationssprache zur Abbildung von Funktionsspezifikationen und Prinziplösungen und ein Vorgehensmodell (Bild 23) umfasst.

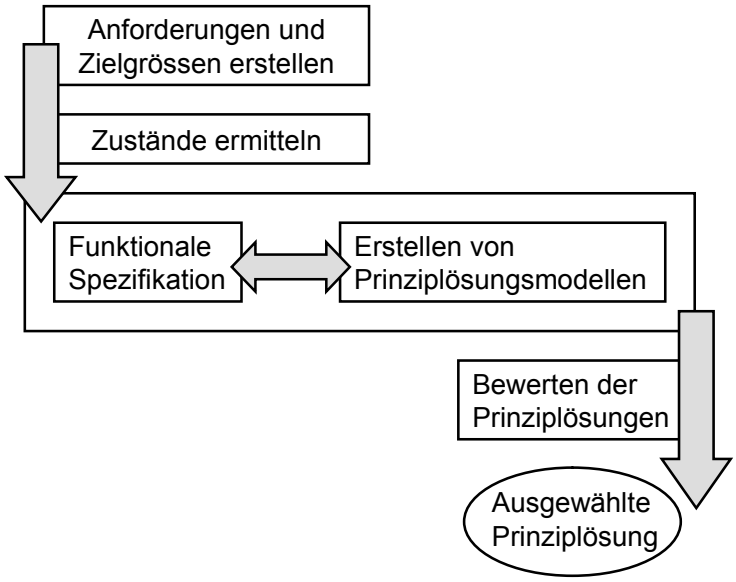


Bild 23: Vorgehensmodell der Produktkonzipierung nach Flath

Nach der Erstellung der Anforderungen und Zielgrössen werden zuerst die Zustände ermittelt in denen sich das Produkt befinden kann. Dabei ist eine Hierarchisierung von Zuständen möglich. In der anschliessenden funktionalen Spezifikation werden Funktionen definiert, die die Übergänge zwischen den Zuständen beschreiben. Diese alternative Verwendung des Funktionsbegriffes berücksichtigt die zeitliche Abfolge von unterschiedlichen Systemzuständen ohne die Ein- und Ausgangsbeziehungen einer Funktion mit Material-, Energie- und Informationsflüssen

zu beschreiben. In der Notation von Flath wird zwischen erwünschten und unerwünschten Zuständen und Funktionen unterschieden.

Funktionsspezifikationen können für verschiedene Zustände eines Systems durchgeführt werden. Dadurch kommt es zu einer redundanten Verwendung von Funktionen und Zuständen. In [27] ist z.B. die Funktionsspezifikation eines Bordnetzes im Zustand „Motor läuft“ und „Motor steht oder startet“ dargestellt. Die Funktion „Elektrische Energie bereitstellen“ und der Zustand „Energiespeicher voll“ ist in beiden Funktionsspezifikationen enthalten. Analog zu den „active purpose functions“ von Buur (Abschnitt 2.4.1) sind je nach Zustand lediglich eine Auswahl aller möglichen Funktionen und Zustände „aktiv“. Analog zu den Zuständen können auch Funktionshierarchien modelliert werden. Die Prinziplösungen werden durch Wirkprinzipien und Lösungselemente nach der Methode von Kallmeyer [43] spezifiziert. Eine Bewertung der Prinziplösungen durch Ermittlung der Zielgrössenerfüllung führt schlussendlich zu der Auswahl einer Prinziplösung.

2.4.5 Molt

In [53] stellt Molt eine Softwarespezifikationstechnik für automatisierte Anlagen vor. Dazu wird das in Bild 24 dargestellte Vorgehen vorgeschlagen.

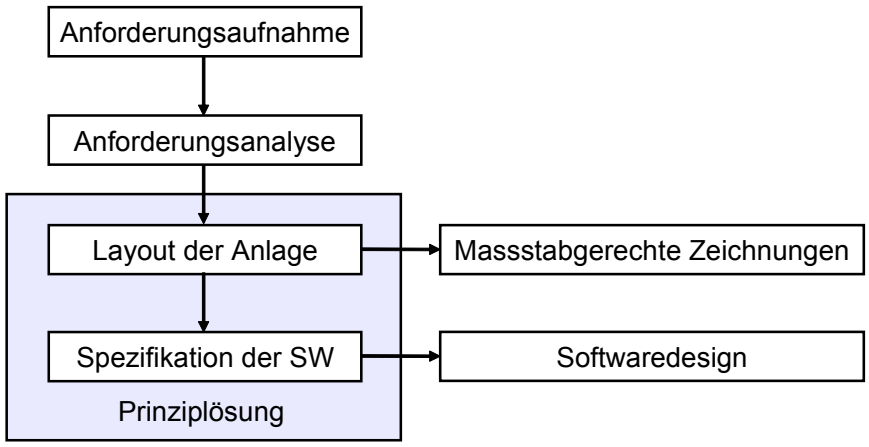


Bild 24: Spezifikation automatisierter Anlagen nach Molt [53]

Die Anforderungen werden aus der Sicht des Anlagenbedieners aufgenommen und als UML-Use-Case-Diagramm [37] dokumentiert. In der Anforderungsanalyse werden die zu projektie-

renden Geräte der Fertigungsanlage bestimmt. Anschliessend wird in dem Layout der Anlage die Platzierung der Geräte skizziert und später in massstabgerechten Zeichnungen dokumentiert. In der Prinziplösung von Molt wird die Softwarespezifikation mit dem skizzierten Layout verknüpft. Dazu führt er Diagramme mit den folgenden Elementen ein:

- *Funktionsobjekte*
  - *Aktionsobjekte*, die eingehende Signale verarbeiten und Ausgangssignale bereitstellen.
  - *Bedingungsobjekte* beinhalten einen Zustand
- *Datenobjekte* beinhalten Daten, die von Aktionsobjekten gesendet oder empfangen werden
- *Ports* spezifizieren die Schnittstellen der Funktionsobjekte
- *Kanäle* transportieren Informationen zwischen den Funktionsobjekten über die Ports

Aktoren und Sensoren werden durch frei wählbare Symbole im Anlagenlayout eingetragen und über Ports mit den Funktionsobjekten verknüpft. Das Konzept der Vererbung aus der objektorientierten Programmierung wird für die oben genannten Objekte unterstützt. Hierarchische Strukturen sind möglich indem Funktionsobjekte rekursiv durch Diagramme mit obigen Elementen verfeinert werden. Alternativ kann die Funktionalität der Funktionsobjekte informell beschrieben werden. Die exakte formale Beschreibung des Verhaltens geschieht anschliessend im Softwaredesign.

Dieser Ansatz verknüpft das informell beschriebene logische Verhalten der Fertigungsanlage mit dessen Layout in der Konzeptphase. Vor der Konzipierung der Software wird die geometrische Ausprägung der Anlage skizziert. Im Fokus sind hier Produktionssysteme und nicht einzelne SPS-gesteuerte Maschinen. Eine Schnittstelle zu einer der fünf genormten SPS-Programmiersprachen [23] ist in dieser Softwarespezifikation nicht vorgesehen.

2.4.6 O^{2}MEN / CAMeL

Im Vordergrund des Objektorientierten Mechatronik Entwurfs (O^{2}MEN) [32] steht das dynamische Verhalten der Bewegung mechatronischer Systeme. Bild 25 zeigt das vorgeschlagene Vorgehen beim Entwurf mechatronischer Systeme.

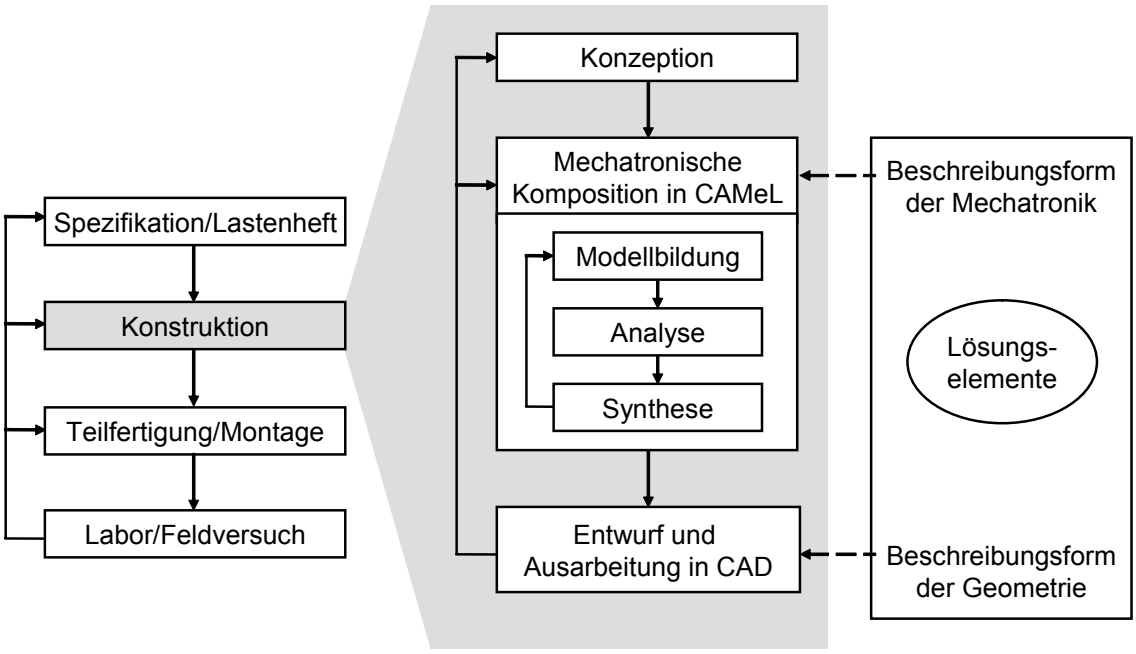


Bild 25: Vorgehensmodell für den Entwurf mechatronischer Systeme in Anlehnung an [6], [32] und [68]

Nach der Formulierung der Anforderungen wird in der Konzeption eine Funktionshierarchie erstellt. Den Funktionen werden Lösungselemente zugeordnet, wobei die Funktionen in drei Funktionstypen aufgeteilt werden:

- Kinematikfunktionen
- Dynamikfunktionen
- Mechatronikfunktionen (Aktorik/Sensorik und Steuerungsfunktionen)

Die verwendete Modellrepräsentation setzt sich aus vier Ebenen zusammen:

- Geometrieebene (CAD)
- Topologiebene (Darstellung der Anordnung und Kopplungen von Bauteilen)
- Verhaltensebene (mathematische Beschreibung des Systemverhaltens)
- Verarbeitungsebene (Codegenerierung zur numerischen Berechnung der Modelle)

Das objektorientierte Mechatronikmodell (OMM) [31] setzt sich aus den letzten drei Ebenen zusammen. Zur integrativen Modellierung, Analyse und Synthese des OMM wurde die Soft-

ware CAMeL (Computer-Aided Mechatronics Laboratory) entwickelt. Die aktuelle Version 6 kann bei der Fima iXtronics [36] bezogen werden. Neben dem OMM lässt sich hier auch die Geometrie des mechatronischen Systems visualisieren (Bild 26).

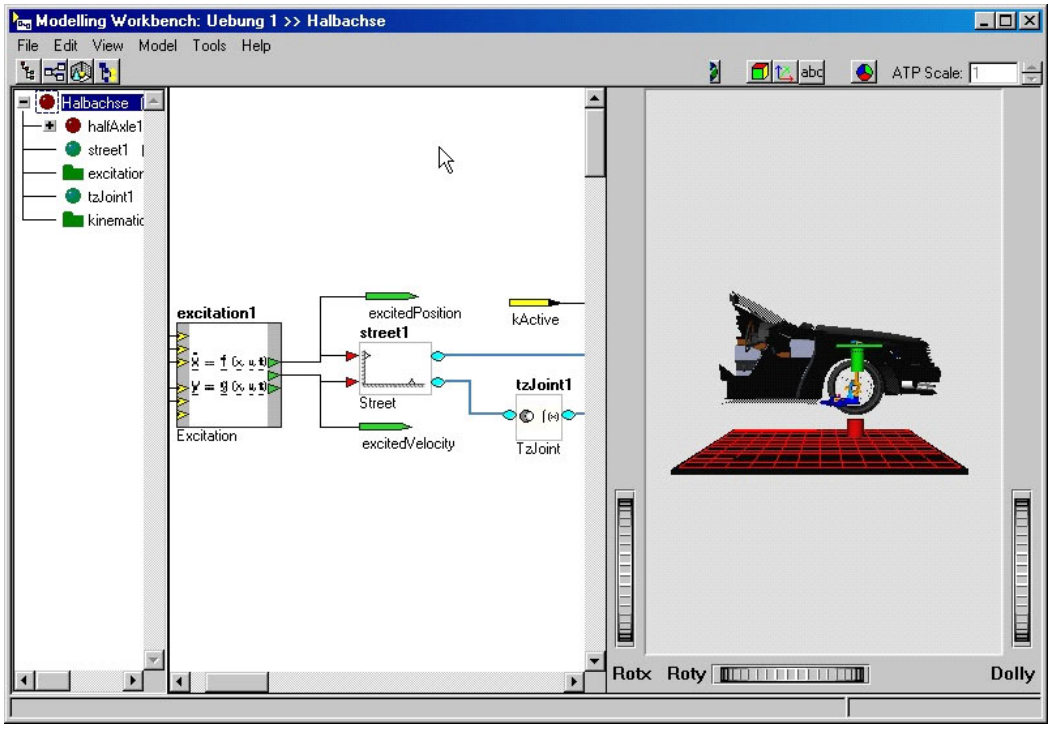


Bild 26: Modellieren mit CAMeL [36]

CAMeL unterstützt den modellbasierten Entwurf mechatronischer Systeme bestehend aus den folgenden drei Phasen:

- Modellphase
- Prüfstandsphase
- Prototypenphase

Nach der Modellbildung unterstütz diese Software SIL (Software-in-the-Loop) und HIL (Hardware-in-the-Loop) Simulationen [72] in der Prüfstandsphase. Für den Bau von Prototypen bietet iXtronics auch Steuerungshardware mit Aktor- und Sensorschnittstellen an.

O^2MEN stellt in Verbindung mit CAMeL ein strukturiertes softwareunterstütztes Vorgehen zur Modellbildung und Untersuchung mechatronischer Systeme dar. Die Konzeptphase im Sinne der etablierten Vorgehensweisen des Maschinenbaus wird nicht durchlaufen [44].

2.4.7 Schemebuilder

Schemebuilder ist ein wissensbasiertes Entwurfswerkzeug für die Konzeption mechatronischer Systeme [55]. Visio [52] kann als Userinterface eingesetzt werden um die „schemes“ (Schemata) zu definieren. Schemebuilder bietet dazu die folgenden Konstrukte an:

- Functions: Funktionen (z.B. Drehmoment aus Strom erzeugen)
- Means: Funktionsträger (z.B. Drehstrommotor)
- Function-means tree (FMT): hierarchische Darstellung der Funktionen und Funktionsträger
- Function-means structure (FMS): ablaufbezogene Darstellung der Funktionen und Funktionsträger
- Working principles: Wirkprinzipien, die über einen FMT in Kombination mit einer FMS beschrieben werden können
- Components: Lösungselemente (z.B. die konkrete Festlegung auf einen Drehstrommotor eines bestimmten Herstellers)

Die obige Aufzählung impliziert das Vorgehen in der Konzeptphase von der Gesamtfunktion bis zur Gesamtlösung bestehend aus Lösungselementen. In Bild 27 ist z.B. das Wirkprinzip eines Servosystems durch einen FMT und eine FMS beschrieben.

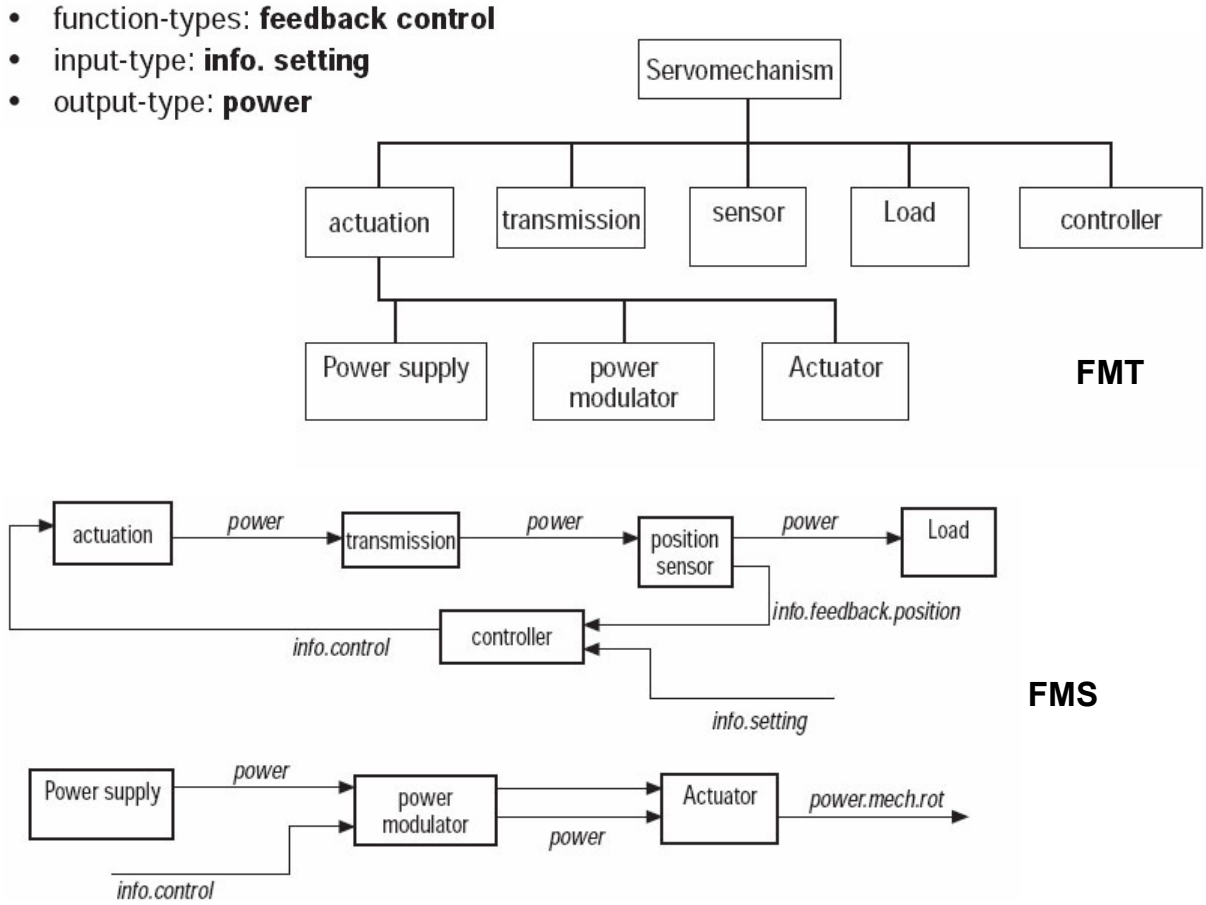


Bild 27: Wirkprinzip eines Servosystems aus [16]

Schemebuilder unterstützt die Suche nach anderen Funktionsträgern einer Funktion. Für die Funktion „Drehmoment aus Strom erzeugen“ wird z.B. auch der Funktionsträger Gleichstrommotor angeboten. Es existieren Visio-Schablonen zur Mechanik, Elektrik und Hydraulik. Desweiteren können generische Bondgraphen modelliert werden. Zur weiteren Analyse kann Schemebuilder MATLAB/Simulink Modelle generieren. Der Export einer der fünf genormten SPS-Programmiersprachen [23] ist in Schemebuilder nicht implementiert.

2.4.8 Process Oriented Analysis (POA)

Die Prozessorientierte Analyse (POA) beruht auf statischen und dynamischen Diagrammen (Flow diagram und State chart), die in Verbindung mit einem strengen Regelwerk als Prozessmodell dienen um ganze Produktionssysteme (Fabriken) zu entwerfen [49]. Die Entwick-

lung einzelner Maschinen steht nicht im Vordergrund. Die Software „POA Toolbox“ (Bild 28) unterstützt die Prozessorientierte Analyse.

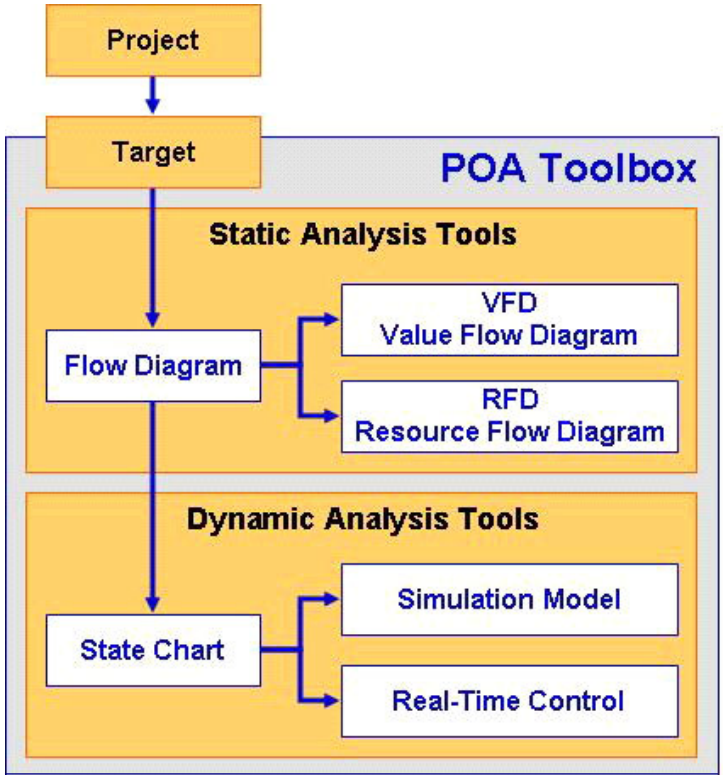


Bild 28: POA Toolbox [60]

Nach der Zielformulierung wird eine Hierachie von Flussdiagrammen erstellt, in denen die Prozesse des Produktionssystems spezifiziert werden. Der Schwerpunkt liegt hier auf den Schnittstellen zwischen den Prozessen mit ihren Material- und Informationsflüssen. Eine Analyse der Kosten kann über das „Value Flow Diagram“ durchgeführt werden, indem zusätzlich Werte und Kosten in das Flussdiagram eingetragen werden. Eine Optimierung der verwendeten Resourcen, wie z.B. Energie, ist durch das „Resource Flow Diagram“ möglich.

Einem Prozess innerhalb eines Flussdiagramms kann ein Zustandsdiagramm (State Chart) zugeordnet werden. Dieses beschreibt alle Zustände und Zustandsübergänge des jeweiligen Prozesses. Basierend auf den Zustandsdiagrammen können nun Simulationsmodelle und Programme für die Steuerung abgeleitet werden. Eine Methode zur Ableitung einer der fünf

SPS-Programmiersprachen [23] existiert nicht. Analog zu den Flussdiagrammen können die Zustandsdiagramme hierarchisch aufgebaut werden (Bild 29).

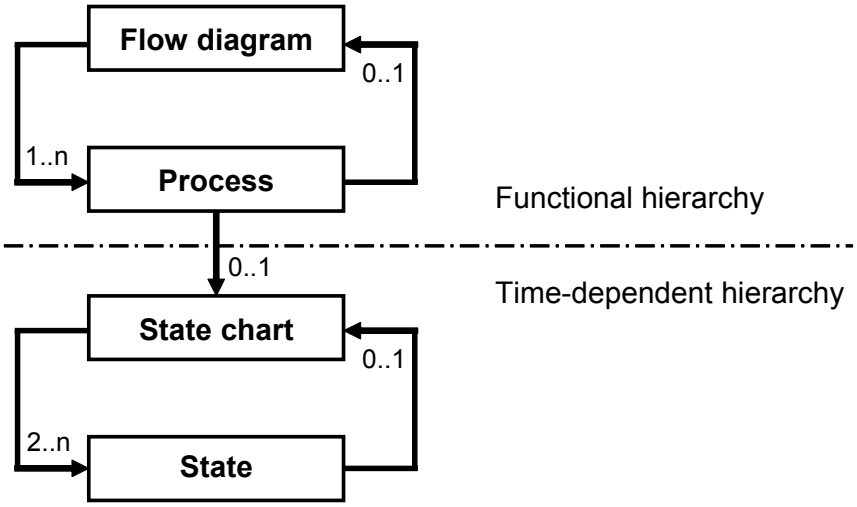


Bild 29: Hierarchie der Prozesse und Zustände in POA

Ein Flussdiagramm enthält mindestens einen Prozess. Einem Prozess kann wiederum ein Flussdiagramm oder ein Zustandsdiagramm zugeordnet werden. Ein Zustandsdiagramm enthält mindestens zwei Zustände, welche wieder ein Zustandsdiagram enthalten können. Durch diese inhaltliche Trennung ergibt sich eine Reihenfolge bei der Betrachtung der funktionalen und zeitabhängigen Aspekte.

2.4.9 Osmers

Osmers schlägt in [57] eine VR-gestützte Projektierung SPS-gesteuerter mechatronischer Systeme vor. Bild 30 zeigt die dazu notwendigen Schritte unter Verwendung einer prototypenhaften Erweiterung des VR-Toolkits der Firma Superscape LTD.

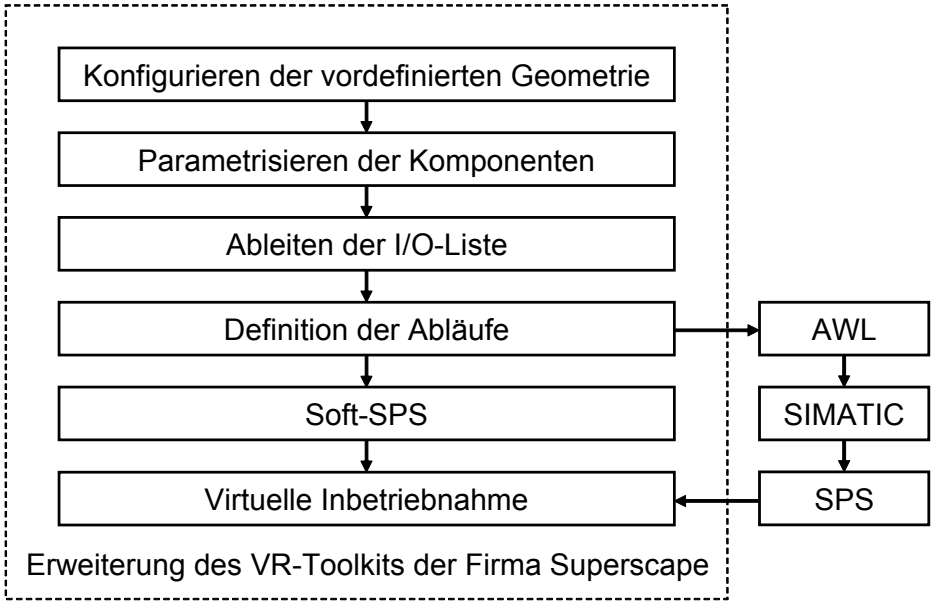


Bild 30: Projektierung und Spezifikation VR-gestützter SPS-Programme nach Osmers [57]

Vordefinierte Geometrie wird aus einer Betriebsmittelbibliothek ausgewählt und in der VR-Umgebung platziert. Dieser Schritt beinhaltet auch das Anbringen vordefinierter Aktoren und Sensoren. Anschliessend werden Parameter, wie z.B. die Stellgeschwindigkeit eines Aktors, in der VR-Umgebung definiert. Wenn alle Aktoren und Sensoren platziert sind, kann die I/O-Liste abgeleitet werden. Unter Verwendung der Variablen der I/O-Liste können die logischen Abläufe in der VR-Umgebung programmiert werden. Eine Soft-SPS innerhalb der VR-Umgebung ermöglicht nun eine erste virtuelle Inbetriebnahme im Stile der Virtuellen Maschine (Bild 16). Es ist auch möglich AWL-Code (Abschnitt 2.3) zu erzeugen, in SIMATIC [65] zu kompilieren und auf eine SPS von SIEMENS zu laden. Über das Multiple Protocol Interface (MPI) von SIEMENS lässt sich die SPS mit der VR-Umgebung koppeln und eine virtuelle Inbetriebnahme mit einer realen SPS durchführen. Dieses Vorgehen setzt einen Konstruktionsentwurf vor der SPS-Programmierung voraus und entspricht somit dem traditionellen Workflow (Bild 2).

2.4.10 eM-PLC

Die Software eM-PLC [69] von Tecnomatix/UGS ist spezifisch auf die Entwicklung SPS-gesteuerter mechatronischer Systeme zugeschnitten (Bild 31).

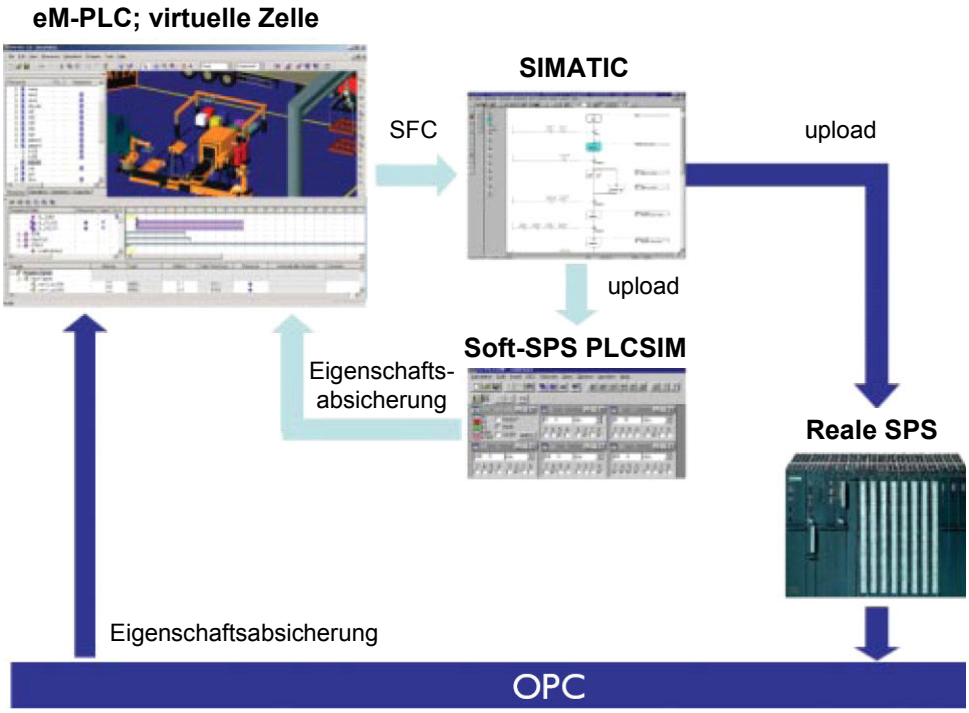


Bild 31: Verwendung des Programms eM-PLC in Kombination mit SIMATIC in Anlehnung an [69].

Nach dem Import des CAD-Modells wird die Kinematik des Systems in eM-PLC definiert. Anschliessend legt der Ingenieur die Ablauflogik durch die sogenannte SOP (sequence of operations) fest. Das Tool eM-PLC ist nun in der Lage das SFC (Abschnitt 2.3) von der SOP automatisch abzuleiten. Anschliessend wird das generierte SFC in die SPS-Programmierumgebung SIMATIC [65] importiert. Ergänzungen am SPS-Programm finden nun je nach Bedarf in SIMATIC statt. Eine erste Eigenschaftsabsicherung (Bild 15) durch eine Virtuelle Maschine (Bild 16) kann durch die Kopplung der Soft-SPS PLCSIM (Bild 31) von SIMATIC mit dem 3D-Modell in eM-PLC geschehen. Schlussendlich wird auch das Erstellen einer zweiten Virtuellen Maschine mit einer realen SPS unterstützt. Dabei kann die im Produkt verwendete SPS über OPC (Open Process Control) mit eM-PLC kommunizieren.

Dieses Vorgehen ist sicherlich für Unternehmen interessant, die noch nicht vom sequentiellen Workflow aus Bild 2 auf den parallelen Workflow aus Bild 4 umstellen wollen. Analog zum Vorgehen von Osmers (Abschnitt 2.4.9) wird die Steuerungstechnik erst nach der Fertigstellung eines CAD-Modells einbezogen.

2.5 Handlungsbedarf

Der Vergleich der Anforderungen aus Abschnitt 1.2 an eine Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme mit dem Stand der Technik ist in Bild 32 zusammengefasst. Die Reihenfolge der Einträge in der Tabelle entspricht der Reihenfolge des Auftretens in dieser Arbeit.

Stand der Technik / Anforderungen	VDI 2221	FOD	VDI 3260, Funktionsdiagr.	DIN EN 60848, GRAFCET	DIN EN 61131-3, SPS-Progr.	VDI 2206	Buur	Stone	Kallenbach	Flath	Molt	O²MEN / CAMeL	Schemebuilder	POA	Osmers	eM-PLC
(A1.1) Beschreibungssprache	✓	✓	✓	✓	✓	✓	✓	✓	✓	✔	✔	✔	✔	✔	–	–
(A1.2) Konzept ➔ Entwurf	–	✓	–	✓	✓	–	–	✓	✓	–	✓	✓	–	✓	–	–
(A1.3) Konsistenter Entwurf	–	–	–	–	–	✓	–	–	✓	–	–	✓	–	✓	✓	✓
(A2) Planmässiges Vorgehen	✔	✔	–	–	✓	✔	–	–	✔	✔	✔	✔	✔	✔	✔	✔
(A3) Softwareunterstützung	–	✔	–	✓	✓	–	–	–	–	–	–	✔	✔	✔	✔	✔

✔ Anforderung erfüllt ✓ Anforderung teilweise erfüllt – Anforderung nicht erfüllt

Bild 32: Bewertung des Stands der Technik im Vergleich zu den Anforderungen

2.5.1 Beschreibung des interdisziplinären Konzeptes durch eine gemeinsame Sprache für SPS-gesteuerte mechatronische Systeme (A1.1)

Damit die Konstruktion gemeinsam mit der Steuerungstechnik ein SPS-gesteuertes mechatronisches System konzipieren kann (Bild 4), muss eine von beiden Domänen akzeptierte, les- und schreibbare Konzeptbeschreibungssprache verwendet werden. Die domänenspezifischen Ansätze aus Abschnitt 2.2 und Abschnitt 2.3 fokussieren naturgemäss entweder auf die Konstruktion oder die Steuerungstechnik. Die von Buur 1989 in [12] geforderte Erweiterung der funktionellen Sicht um den Aspekt Zeit wurde in einigen Ansätzen erreicht indem die funktio-

nelle Sicht mit einer zeitabhängigen Darstellung (hierarchisch) verknüpft wurde. So werden z.B. nach Flath (Abschnitt 2.4.4) zuerst die Zustände des Systems ermittelt, um anschliessend die Zustandsübergänge durch Funktionen zu beschreiben. In der POA (Abschnitt 2.4.8) werden zuerst die Prozesse in einer funktionellen Hierarchie beschrieben. Zustandsdiagramme können dann einem Prozess zugeordnet werden (Bild 29). Die Trennung zwischen zeitabhängigen und zeitunabhängigen Aspekten bleibt bestehen. Eine direkte Unterstützung SPS-gesteuerter mechatronischer Systeme bietet keiner der Ansätze.

2.5.2 Reibungsloser Übergang vom gemeinsam erstellten Konzept zum Steuerungs- und Konstruktionsentwurf (A1.2)

In Abschnitt 2.2 und Abschnitt 2.3 sind bewährte Ansätze beschrieben, um innerhalb einer Domäne von einem domänenspezifischen Konzept den domänenspezifischen Entwurf im CAD oder der SPS-Programmierumgebung zu starten. Existierende interdisziplinäre Ansätze für die weitere Verwendung einer Konzeptbeschreibung fokussieren oft auf den reibungslosen Übergang vom Konzept zur Modellbildung und -analyse (Bild 33).

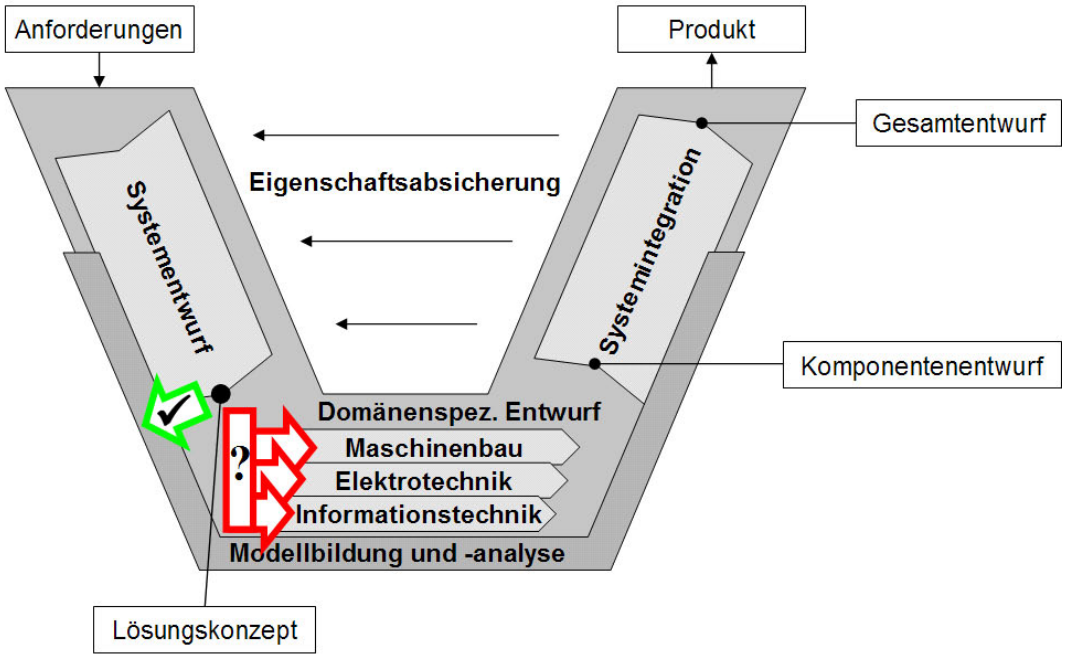


Bild 33: Übergang vom Lösungskonzept zum domänenspezifischen Entwurf

Der Übergang vom Konzept zum domänenspezifischen Entwurf bedingt eine Extraktion der domänenspezifischen Daten, so dass die domänenspezifischen Entwurfswerkzeuge eingesetzt werden können. Dies deckt sich mit der Forderung von Kallenbach (Abschnitt 2.4.3) an ein mechatronisches Entwufssystem, in der die domänenspezifische Entwurfssoftware methodisch und mittels Softwareschnittstellen eingebettet ist. Typisch für diese Problematik ist der Schemebuilder-Ansatz (Abschnitt 2.4.7). Nachdem die Funktionen und deren Umsetzung in Schemebuilder definiert wurden, können automatisch MATLAB/Simulink Modelle generiert werden. Der Einstieg in die domänenspezifischen Entwurfswerkzeuge wird nicht unterstützt.

2.5.3 Verbesserung der Konsistenz interdisziplinär relevanter Daten im domänenspezifischen Entwurf (A1.3)

Der Handlungsbedarf, die domänenübergreifende Konsistenz beim concurrent engineering zu gewährleisten, besteht ganz allgemein für alle mechatronischen Systeme [62], [72]. Wenn die interdisziplinäre Konsistenz zwischen der Steuerungstechnik und der Konstruktion thematisiert wird, geschieht dies in der Regel iterativ. Zuerst wird der Konstruktionsentwurf erstellt, anschliessend das SPS-Programm. So kann z.B. mit der Software eM-PLC (Abschnitt 2.4.10) durch eine Verknüpfung der SPS mit der Geometrie die interdisziplinäre Konsistenz des Gesamtentwurfs überprüft werden. Insbesondere wird eine Einigkeit zwischen der Steuerugstechnik und der Konstruktion über die verwendeten Aktoren und Sensoren auf der elektromechanischen Grenzlinie (Bild 5) erzwungen. Der gleichzeitige Start der Domänen mit einem kontinuierlichen Abgleich, falls z.B. die Steuerungstechnik innerhalb der Ablauflogik einen zusätzlichen Sensor verwendet, wird nicht unterstützt.

Innerhalb der Domänen existiert schon Software zur Wahrung der Konsistenz innerhalb der Konstruktion und Steuerungstechnik. Die Verknüpfung eines Produktdatenmanagement-Systems (wie z.B. das PDM-System Teamcenter von UGS [70]) mit einem CAD-System erlaubt es, innerhalb der Konstruktion Änderungen des CAD-Modelles anderen Konstrukteuren zur Verfügung zu stellen. Durch die Vergabe von Lese- und Schreibrechten für die einzelnen Baugruppen lassen sich Konflikte durch redundante Änderungen vermeiden und geometrische Constraints an benachbarte Baugruppen definieren. Analog lassen sich mit Managementsystemen für Quellcode (wie z.B. VersionWorks for Automation [28] oder Visual SourceSafe [51]) in Verbindung mit SPS-Programmierumgebungen (wie z.B. SIMATIC [65] oder B&R Automation Studio [7]) Änderungen in der Software verwalten. Steuerungstechniker können für die

verschiedenen Softwaremodule Lese- oder Schreibrecht zugewiesen bekommen. Änderungen in der Software können so konsistent publiziert werden. Arbeiten zwei Steuerungstechniker am selben Modul, werden die Änderungen entweder konsistent fusioniert oder das inkonsistente Codefragment wird vor dem „check in“ angezeigt.

2.5.4 Planmässiges Vorgehen zur Entwicklung SPS-gesteuerter Systeme (A2)

Das planmässige Vorgehen ist ein wesentlicher Bestandteil jeder Entwicklungsmethodik [59]. So strukturieren die meisten Ansätze auch den Ablauf innerhalb des Entwicklungsprozesses bezogen auf ihren Beitrag zur Verbesserung der Entwicklung mechatronischer Systeme. Die Richtlinie VDI 2206 [72] bietet hierzu einen guten Rahmen zur weiteren Detaillierung und Spezialisierung des Vorgehens durch die Erweiterung existierender und/oder Formulierung zusätzlicher vordefinierter Prozessbausteine (Bild 17). In der VDI 2206 sind Prozessbausteine für die Arbeitsschritte Systementwurf, Modellbildung und -analyse, domänenspezifischer Entwurf, Systemintegration und Eigenschaftsabsicherung beschrieben. Allerdings ist die notwendige Aufteilung der Funktionserfüllung unter den beteiligten Domänen (Partitionierung) lediglich erwähnt. Ein planmässiges Vorgehen zur Erreichung dieser Partitionierung wird nicht angegeben. Analog dazu findet sich auch in den anderen Ansätzen kein planmässiges Vorgehen zur Partitionierung des interdisziplinären Konzeptes für das CAD und die SPS-Programmierumgebung.

2.5.5 Unterstützung der Konstruktion und Steuerungstechnik durch Software (A3)

Dem grossen Einfluss von Entscheidungen auf das geplante Produkt in der Konzeptphase stehen wenig bis keine Softwarelösungen gegenüber [78]. Dieser Trend kehrt sich bei fortschreitender Produktreife um. Es stehen dann viele Tools zur Verfügung, die Möglichkeiten der Einflussnahme auf das Produkt nehmen dafür ab. Insbesondere in der frühen Konzeptphase, in der Anforderungen auf funktionale Beschreibungen abgebildet werden, besteht ein grosses Manko an softwaretechnischer Unterstützung [78]. FOD (Bild 11) bietet hier eine durchgängige Unterstützung durch Software im Entwurf. Allerdings beschränkt sich FOD auf die Domäne Konstruktion. Die domänenübergreifende Software von Osmers (Abschnitt 2.4.9) und eM-PLC (Abschnitt 2.4.10) greifen erst ab dem Konstruktionsentwurf. Die drei verbleibenden Ansätze

in Bild 32 mit Softwareunterstützung (CAMeL-Abschnitt 2.4.6, Schemebuilder-Abschnitt 2.4.7 und POA-Abschnitt 2.4.8) bieten keine Schnittstelle zu einer SPS-Programmierumgebung.

2.5.6 Fazit

Insgesamt ergibt die Analyse der vorhandenen Ansätze zur Entwicklung SPS-gesteuerter mechatronischer Systeme in diesem Kapitel, dass keiner der Ansätze die in Abschnitt 1.2 formulierten Anforderungen erfüllt. Somit besteht Handlungsbedarf diese Forschungslücke durch Methoden zu schliessen, die in ein planmässiges Vorgehen eingebettet sind und durch Software unterstützt werden.

3 Interdisziplinäre Entwicklung SPS-gesteuerter mechatronischer Systeme

Ziel dieser Arbeit ist es die interdisziplinäre Entwicklung SPS-gesteuerter mechatronischer Systeme zu verbessern. Insbesondere soll die gemeinsame Konzeption inklusive des nachfolgenden domänenspezifischen Entwurfs erleichtert werden. In diesem Kapitel wird dazu ein Ansatz zur Erfüllung der Anforderungen (A1)-(A3) aus Abschnitt 1.2 vorgestellt (Bild 34), der die in Abschnitt 2.5.6 festgestellte Forschungslücke schliesst.

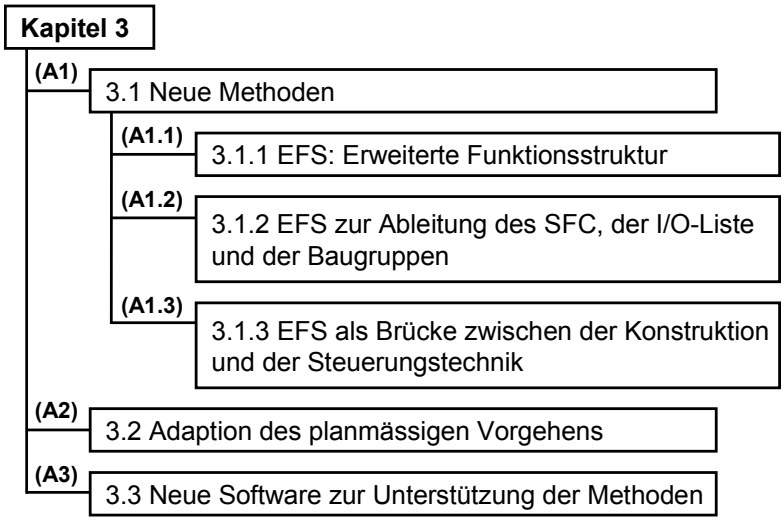


Bild 34: Die Anforderungen (A1)-(A3) aus Abschnitt 1.2 in Relation zu den Abschnitten in diesem Kapitel

Die Struktur der Abschnitte in diesem Kapitel ergibt sich aus den Anforderungen und spiegelt die Definition der Entwicklungsmethodik nach Pahl-Beitz [59]. Neue Methoden zur Beschreibung und Nutzung der interdisziplinären Konzeption SPS-gesteuerter mechatronischer Systeme sind im nächsten Abschnitt 3.1 beschrieben. Unter Verwendung der aktuellen Entwicklungsrichtlinie für mechatronische Systeme [72] werden diese Methoden anschliessend im Abschnitt 3.2 in ein planmässiges Vorgehen eingebettet. Zur Unterstützung der hier vorgestellten Methodik beschreibt der letzte Abschnitt 3.3 eine Software, welche im Rahmen dieser Dissertation entstanden ist.

3.1 Neue Methoden

In Abschnitt 3.1.1 wird die Erweiterte Funktionsstruktur (EFS) vorgestellt, welche es erlaubt, auch Aspekte der Steuerungstechnik schon in einer frühen Phase der Produktentwicklung mit einfliessen zu lassen. Die Methode in Abschnitt 3.1.2 zeigt anschliessend, wie aus der EFS das SFC und die I/O-Liste (Abschnitt 2.3) abgeleitet werden kann. Zusätzlich ist es möglich, die Baugruppen für den Einstieg ins CAD-Modellieren aus der EFS in Anlehnung an Stone [67] abzuleiten. In Abschnitt 3.1.3 wird die EFS genutzt, um interdisziplinär relevante Änderungen zu managen.

3.1.1 EFS: Erweiterte Funktionsstruktur

Die Erweiterung der Funktionsstruktur besteht im Wesentlichen aus Übergangsbedingungen, wie sie von Petrinetzen oder dem SFC aus Abschnitt 2.3 bekannt sind. Informationsflüsse können in der EFS Übergangsbedingungen enthalten. So kann in der ablaufbezogenen Darstellung der Funktionsstruktur (Abschnitt 2.2) mit Hilfe dieser Erweiterung ausgedrückt werden, dass eine Funktion erst nach einer definierten Übergangsbedingung relevant wird. Wird z.B. eine Flaschenabfüllanlage modelliert, besteht dann die Möglichkeit vor der Funktion „Flasche füllen“ die Übergangsbedingung „Flasche in Abfüllposition“ zu definieren. Mit Hilfe dieser Übergangsbedingungen wird die typische Sensorik (Abschnitt 2.3) einer SPS abgebildet. Der dazugehörige Input (Abschnitt 2.3) für die SPS ist oft sofort klar und kann dann auch der Übergangsbedingung zugeordnet werden. So kann man im Beispiel der Flaschenabfüllanlage der Übergangsbedingung „Flasche in Abfüllposition“ direkt die Inputvariable „diBottleInFillPosition“ (Abschnitt 2.3) zuordnen. Sobald definiert ist, welcher Sensor zum Einsatz kommt, kann auch dieser der Übergangsbedingung hinterlegt werden.

Aktoren, die von einer SPS angesteuert werden, erhalten von ihr Informationen (Bild 5). Diese Aktoren wandeln den Informationen entsprechend ihre Versorgungsenergie in eine Energie um, die auf das Grundsystem wirkt. Dies spiegelt sich auch in der Funktionsstruktur wieder. Funktionen, die solch einen Aktor repräsentieren, haben mindestens einen Informationsfluss und einen (Versorgungs-)Energiefluss als Eingang. Mindestens ein Energiefluss wirkt von der Aktorfunktion auf eine andere Funktion des Grundsystems. Solche Funktionen werden in dieser Arbeit „Aktorfunktion“ genannt. Sie sind Elementarfunktionen (Abschnitt 2.2) der EFS, da die entsprechenden Aktoren eingekauft werden und nicht mehr im Detail betrachtet werden müssen

(Abschnitt 2.4). Dadurch ergibt sich als eine Abbruchbedingung für die rekursive top-down Modellierung der hierarchischen Funktionsstruktur das Auftreten von Aktorfunktionen. Analog zu den Sensoren können hier auch die Outputvariablen für die SPS und der spezifische Aktor definiert und der Aktorfunktion zugeordnet werden. Im Beispiel der Flaschenabfüllanlage kann die Funktion „Flasche füllen“ eine Unterfunktion „Flüssigkeit für Flasche freigeben“ haben. Diese Unterfunktion stellt in diesem Fall gleichzeitig eine Aktorfunktion dar. Hier kann z.B. ein Ventil mit einer boolschen Outputvariable „doVentilOpen“ zugeordnet und auf „1“ gesetzt werden. Das Beschreiben des Konzeptes unter Verwendung der EFS läuft insgesamt wie folgt ab:

- Zuerst wird die Gesamtfunktion (Abschnitt 2.2) des mechatronischen Systems definiert.
- Anschliessend ergibt sich die Funktionshierarchie durch das rekursive Definieren von Unterfunktionen. Unterfunktionen von Aktorfunktionen werden nicht mehr angelegt.
- Die entsprechenden Material-, Energie- und Informationsflüsse werden zwischen den Funktionen eingetragen. Auch die Übergangsbedingungen können nun festgehalten werden.
- Abschliessend folgt die Zuordnung der entsprechenden Input- und Outputvariablen sowie der jeweiligen Sensoren und Aktoren zu ihren Aktorfunktionen und Übergangsbedingungen.

Die Erstellung der EFS erstreckt sich bei einer Neuentwicklung über einen längeren Zeitraum. So müssen parallel zur Definition der Unterfunktionen auch über Lösungsprinzipien nachgedacht werden, wie die Funktionen zu erfüllen sind ([72], [73] und [59]). Dadurch wird eine noch lösungsneutrale Gesamtfunktion durch das Anwachsen der Funktionshierarchie immer konkreter. Die oben implizierte Reihenfolge muss zudem nicht immer zwingend eingehalten werden. Die Flüsse der Gesamtfunktion können z.B. schon definiert werden, ohne dass die Funktionshierarchie besteht. Auch Übergangsbedingungen können oft schon zwischen Hauptfunktionen (Abschnitt 2.2) eingetragen werden. Bei der Definition der Flüsse und der Übergangsbedingungen hat es sich bewährt, in der folgenden Reihenfolge vorzugehen:

1. Materialflüsse
2. Energieflüsse
3. Informationsflüsse
4. Übergangsbedingungen

Insgesamt beschreibt die EFS die funktionalen Zusammenhänge im gesamten mechatronischen System und besteht aus den folgenden Elementen:

- Funktionen (analog zur FS: Bild 8)
- Material-, Energie- und Informationsflüsse zwischen den Funktionen (analog zur FS)
- Hierarchie zwischen den Funktionen (analog zur FS)
- Übergangsbedingungen auf den Informationsflüssen (neu in der EFS)
- Aktor- und Sensordefinitionen (neu in der EFS)
- definierte Input- und Outputvariablen (neu in der EFS)

Die Ablauflogik der SPS ist über die Informationsflüsse und den Events der Inputs (via Übergangsbedingung) und Outputs (via Aktorfunktion) abgebildet. Die spezifizierten Aktoren und Sensoren mit den zugeordneten Output- und Inputvariablen sind hinterlegt. Schlussendlich sind die funktionellen Zusammenhänge des Grundsystems (Bild 5) aus den traditionellen Elementen der EFS ersichtlich.

Im domänenspezifischen Entwurf verwendet jede Domäne ihre eigenen Werkzeuge: hauptsächlich CAD in der Konstruktion und eine SPS-Programmierumgebung in der Steuerungstechnik. Wie nun aus der interdisziplinären Beschreibung des mechatronischen Systems durch die EFS die relevanten Informationen für den Entwurf in den beiden Domänen Konstruktion und Steuerungstechnik abgeleitet werden können, wird im nächsten Abschnitt beschrieben.

3.1.2 EFS zur Ableitung des SFC, der I/O-Liste und der Baugruppen

Beim Programmieren der SPS in der Steuerungstechnik sind lediglich die Informationsflüsse (Bild 5) der EFS relevant. Die Ablauflogik des Systems ist in der EFS über diese Informationsflüsse modelliert. Streicht man nun alle Funktionen in der EFS, die lediglich Material- und Energieflüsse haben, erhält man eine Beschreibung dieser Ablauflogik [4]. Durch die Verwendung der entsprechenden I/O‘s ergibt sich wie in Bild 35 dargestellt ein SFC (Abschnitt 2.3):

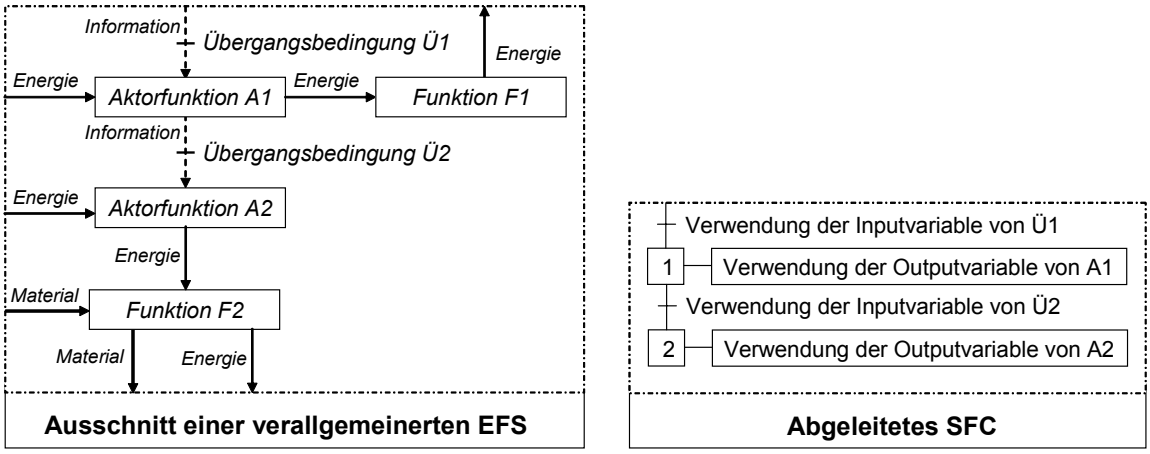


Bild 35: Ableitung eines SFC aus einer EFS

SFC eignet sich gut zur konzeptionellen Darstellung der Ablauflogik (vgl. auch Bild 14). Damit lassen sich die wesentlichen Aspekte zur Steuerung des Systems über die SPS abbilden. Da SFC im Gegensatz zur EFS eine Programmiersprache ist, lässt es sich kompilieren und auf eine SPS laden. Typischerweise sind aber nicht alle Aspekte für die Steuerungstechnik in der EFS abgebildet. So muss z.B. oft noch ein Mensch/Maschinen Interface programmiert werden. Manche Überwachungsfunktionen ergeben sich erst beim Programmieren. Auch das Fehlermanagement für Fehler innerhalb und ausserhalb der Software, ist in der EFS noch nicht zwingend definiert. Über die EFS soll der Steuerungstechnik via SFC die grundsätzliche Ablauflogik zur Verfügung stehen. Basierend auf dem SFC kann dann, wie in Bild 14 dargestellt, die SPS ausprogrammiert werden. Dazu können alle fünf IEC 1131-3 Sprachen [23] kombiniert zum Einsatz kommen.

Zusätzlich muss die Steuerungstechnik wissen, über welche Input- und Outputvariablen sie die jeweiligen Aktoren und Sensoren ansprechen kann. Dies ist in der I/O-Liste (Abschnitt 2.3) festgehalten. Durch die Auflistung aller I/O‘s der EFS kann nun der Steuerungstechnik auch die I/O-Liste zur Verfügung gestellt werden.

Die zweite wichtige Domäne beim Entwurf SPS-gesteuerter mechatronischer Systeme ist die Konstruktion (Kapitel 2). Mit Hilfe des CAD wird hier die Geometrie des elektromechanischen Subsystems bestehend aus Aktoren, Sensoren und dem Grundsystem modelliert (Bild 5). Ausgehend von der EFS können über die Heuristik von Stone (Abschnitt 2.4.2) Baugruppen iden-

tifiziert werden (Bild 36). Bei der Suche nach Lösungsprinzipien für die Funktionen der EFS bietet der Morphologische Kasten (Abschnitt 2.2) Unterstützung.

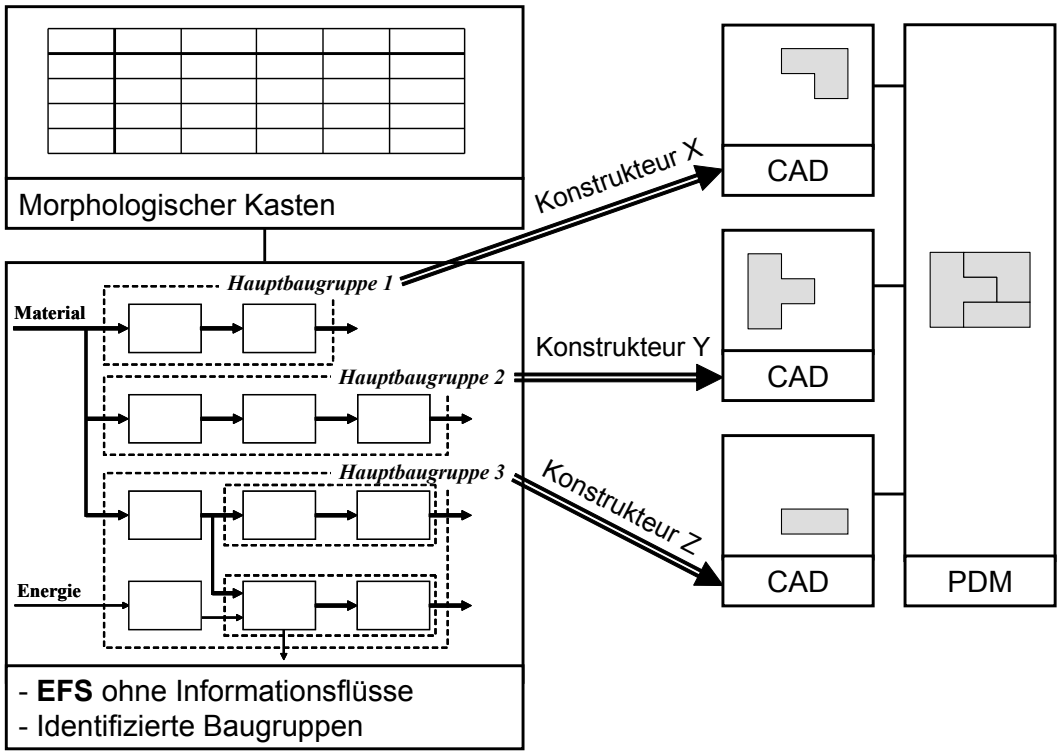


Bild 36: Die EFS als Ausgangspunkt für den Konstruktionsentwurf

Nach Stone [67] werden die Material- und Energieflüsse der EFS bezüglich der drei Regeln „Dominant flow“, „Branching flow“ und „Conversion-transmission“ hin untersucht. Die ermittelten Hauptbaugruppen können dann einzelnen Konstrukteuren zugeordnet werden. Zusätzlich lassen sich alle Aktoren und Sensoren ihren Hauptbaugruppen über die EFS zuordnen. Dadurch weiss der Konstrukteur schon vor dem CAD-Entwurf welche Aktorik und Sensorik in „seiner“ Hauptbaugruppe verbaut werden soll. Nicht nur für das „concurrent engineering“ innerhalb der Konstruktion, sondern auch für das top-down Modellieren mittels CAD ist die Identifikation der Hauptbaugruppen wichtig, um den Konstruktionsentwurf zu starten. Typischerweise werden alle CAD-Modelle in einem PDM-System so verwaltet, dass alle Konstrukteure Zugriff auf den jeweils aktuellen publizierten Stand aller Baugruppen des Projektes haben.

3.1.3 EFS als Brücke zwischen der Konstruktion und der Steuerungstechnik

Wie in Abschnitt 2.5 beschrieben, sind beim Entwurf mechatronischer Systeme vor allem Änderungen problematisch, die nicht nur für eine Domäne relevant sind, sondern sich domänenübergreifend auswirken. Aktoren und Sensoren stellen ein Bindeglied zwischen der SPS-Software der Steuerungstechnik und dem CAD-Modell des Grundsystems der Konstruktion dar (Bild 5). Die Geometrie der Aktoren und Sensoren wird von der Konstruktion im CAD modelliert. Aus Sicht der Steuerungstechnik interessieren die korrespondierenden Input- und Outputvariablen. Über die Inputvariablen lassen sich die Sensorinformationen im Steuerungscode adressieren. Die Aktoren werden über die Outputvariablen angesteuert. Änderungen dieser interdisziplinär relevanten Daten in der Konstruktion oder der Steuerungstechnik betreffen jeweils auch die andere Domäne. Führt man diese Änderung in der EFS nach, kann man feststellen, inwieweit die andere Domäne betroffen ist (Bild 37).

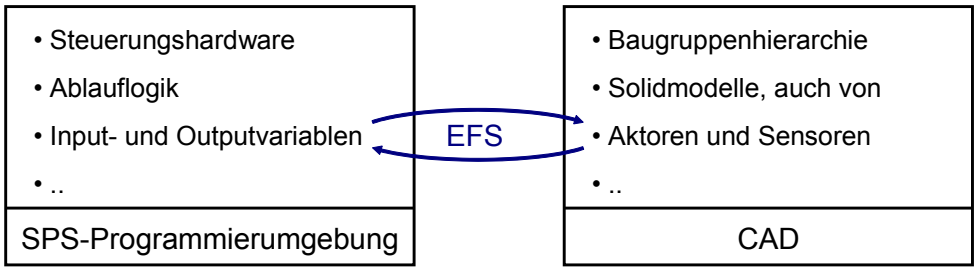


Bild 37: Änderung interdisziplinärer Daten via EFS

Einen präziseren Überblick über die wichtigsten Entitäten im interdisziplinären Änderungswesen verschafft das Bild 38. Die Abhängigkeiten zwischen diesen Entitäten sind in diesem Strukturdiagramm nach UML 2 ([56], [37]) dargestellt.

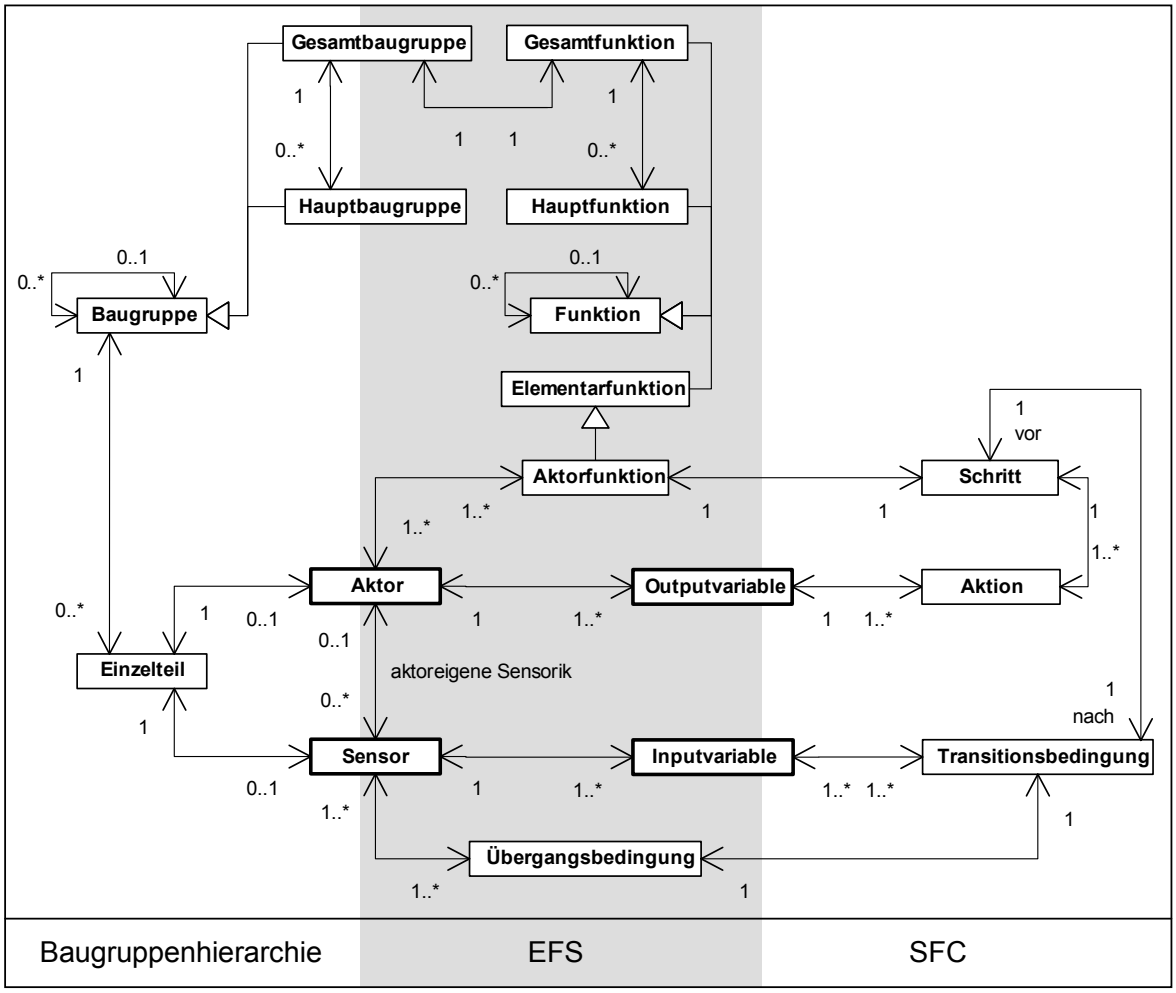


Bild 38: Strukturdiagramm nach UML 2 für die wichtigsten Entitäten und Beziehungen im interdisziplinären Änderungswesen

Innerhalb der Baugruppenhierarchie des CAD sind die Baugruppen analog zu den Funktionen hierarchisch strukturiert. Die Gesamtbaugruppe stellt die elektromechanischen Funktionaliäten der Gesamtfunktion zur Verfügung. Unterhalb der Gesamtbaugruppe befinden sich die Hauptbaugruppen, welche als Baugruppe rekursiv wieder Baugruppen oder am Ende der Baugruppenhierarchie Einzelteile enthalten. Ausser der 1:1 Relation zwischen der Gesamtbaugruppe und der Gesamtfunktion kann innerhalb der Hierachie die Beziehung zwischen den Baugruppen und Funktionen nicht a priori festgelegt werden. Einige der Einzelteile eines SPS-gesteuerten mechatronischen Systems sind Aktoren oder Sensoren. Über Aktorfunktionen und Übergangsbedingungen sind diese Einzelteile in der EFS inklusive ihrer Output- und Inputvariablen dokumentiert. Da die Funktionen innerhalb der Aktorfunktion nicht weiter diskutiert werden,

ist die Aktorfunktion eine Elementarfunktion (Abschnitt 3.1.1). Jeder Aktorfunktion ist ein Schritt (Bild 13) im SFC zugeordnet. Das in der Aktorfunktion definierte Verhalten des Aktors impliziert die Werte der Outputvariablen für die Aktionen innerhalb des Schrittes im SFC. Die Transitionsbedingungen inklusive der verwendeten Inputvariablen entsprechen den Übergangsbedingungen in der EFS.

Die Aktoren und Sensoren mit ihren Output- und Inputvariablen stellen den Dreh- und Angelpunkt zur Propagation interdisziplinär relevanter Änderungen dar. Aktor- und Sensordefinitionen finden sich sowohl in der EFS als auch in der Baugruppenhierarchie der Konstruktion. Die Output- und Inputvariablen werden neben der EFS auch im SFC und der I/O-Liste in der Steuerungstechnik verwendet. Die Zukaufteile Aktoren (Abschnitt 2.4) können selbst wiederum Sensoren zur Überwachung des Aktorstatus beinhalten. So enthalten Drehmotoren oft einen Encoder [5] zur Messung der aktuellen Drehzahl. Für diese aktoreigene Sensorik ist in Bild 38 eine Beziehung zwischen der Entität Aktor und der Entität Sensor modelliert. Eine Änderung, die sich auf einen aktoreigenen Sensor auswirkt, hat somit auch auf den Aktor einen entsprechenden Einfluss. Änderungen können sein:

- Neudefinition einer Entität
- Ändern einer Entität
- Löschen einer Entität

Für jeden der drei Fälle wird im Folgenden ein Beispiel skizziert. Zuerst wird dabei eine Änderung im SFC beschrieben. Dann folgt ein Beispiel wo von einer Änderung in der EFS ausgegangen wird. Schlussendlich wird der Einfluss einer interdisziplinär relevanten Änderung in der Baugruppenhierarchie durch die Konstruktion dargelegt.

Neue Inputvariable im SFC:
Wird in der Steuerungstechnik eine neue Inputvariable zur Überwachungen eines bestimmten Systemverhaltens angelegt und in einer Transitionsbedingung verwendet, so führt dies auch zu einer neuen Inputvariablen inklusive Übergangsbedingung und Sensor in der EFS. Der Sensor muss nun ebenfalls als Einzelteil in der Baugruppenhierarchie vorhanden sein.

Ändern einer Aktorfunktion in der EFS:
Die EFS kann durch die Steuerungstechnik oder die Konstruktion angepasst werden. In diesem Szenario ist ein Drehmotor, der links und rechts herum laufen kann, einer Aktorfunktion so hinterlegt, dass er startet und rechts läuft. Wird nun in der Aktorfunktion anstelle des Rechtslaufs definiert, dass der Motor hier links herum laufen soll, beeinflusst dies in der EFS lediglich das Setzen der Outputvariablen. Der Aktor mit seinen Variablen verbleibt unverändert in der EFS, so dass weder die I/O-Liste noch die Baugruppenhierarchie geändert werden müssen. Allerdings ändern sich die Aktionen im SFC, da nun die Outputvariablen mit anderen Werten belegt sind.

Löschen eines Sensors in der Baugruppenhierarchie:
Entscheidet sich die Konstruktion für das Entfernen eines Sensors, wird dies durch ein Löschen des entsprechenden Einzelteils in der Baugruppenhierarchie und der EFS reflektiert. In der EFS führt dies zur Beseitigung aller Inputvariablen des Sensors. Eine Aktualisierung der I/O-Liste (Abschnitt 3.1.2) in einer SPS-Programmierumgung (wie z.B. von B&R [7] oder SIEMENS [65]) generiert spezifische Fehlermeldungen beim Kompilieren des Quellcodes. Um wieder einen konsistenten Gesamtentwurf zu erreichen, muss der Steuerungstechniker nun überall dort aktiv werden, wo der Kompiler die nunmehr ungültige Verwendung der gelöschten Inputvariablen im Quellcode anzeigt.

In diesem Abschnitt wurden die Beziehungen zwischen den Entitäten der Baugruppenhierarchie, der EFS und des SFC beschrieben. Zusätzlich wurden anhand von Bild 38 die Auswirkungen diskutiert, falls eine der Entitäten erzeugt, geändert oder gelöscht wird. Zur konsistenten Verwaltung der Entitäten bietet sich eine Unterstützung durch Software an. Dort lassen sich dann unter Verwendung von Bild 38 die Implikationen, die sich bei Änderungen ergeben, automatisieren. In Abschnitt 3.3 wird eine neue Software vorgestellt, die neben der Definition der EFS (Abschnitt 3.1.1) und dessen Nutzung (Abschnitt 3.1.2) auch deren konsistente Verwaltung ermöglicht.

3.2 Adaption des planmässigen Vorgehens

Der angestrebte Workflow, wie er in Bild 4 dargestellt ist, deckt sich mit den Zielen der Entwicklungsrichtlinie VDI 2206 für mechatronische Systeme [72]. Wie aus Bild 15 ersichtlich,

wird auch in der VDI 2206 ein gemeinsamer Systementwurf zur Erstellung des Konzeptes und ein paralleles Arbeiten im domänenspezifischen Entwurf vorgeschlagen. Durch das Eintragen der beiden Hauptentwurfswerkzeuge CAD und SPS-Programmierumgebung (Abschnitt 2.4) wird aus dem V-Modell der VDI 2206 (Bild 15) das in Bild 39 dargestellte V-Modell [1].

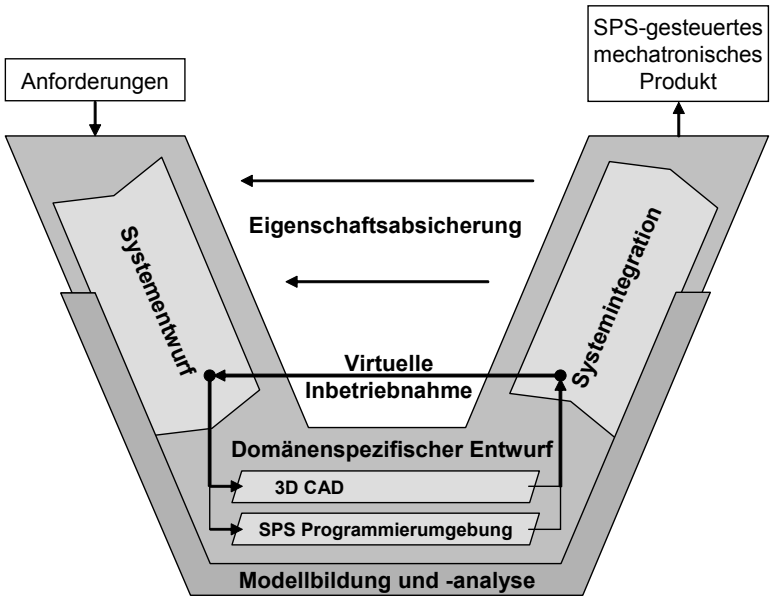


Bild 39: Angepasstes V-Modell für SPS-gesteuerte mechatronische Systeme [1]

In diesem Abschnitt wird aufgezeigt, wie durch die Verwendung der EFS (Abschnitt 3.1.1) im Systementwurf ein planmässiger Übergang in den domänenspezifischen Entwurf bewerkstelligt werden kann. Dazu wird ein neuer vordefinierter Prozessbaustein (Abschnitt 2.4) für die Domänenverzweigung in Bild 41 vorgeschlagen. Zusätzlich wird zur Eigenschaftsabsicherung über eine virtuelle Inbetriebnahme durch die Virtuelle Maschine (Abschnitt 2.4) ein neuer vordefinierter Prozessbaustein präsentiert, der einen planmässigen Übergang vom domänenspezifischen Entwurf zur Systemintegration beschreibt.

Für den Systementwurf gibt es in der VDI 2206 schon einen vordefinierten Prozessbaustein (Bild 17). Die erste Anpassung ergibt sich durch die Verwendung der EFS anstelle der Funktionsstruktur aus der Konstruktionslehre. Die zweite Anpassung in Bild 40 ergibt sich durch den Einschub der Domänenverzweigung nach dem Systementwurf.

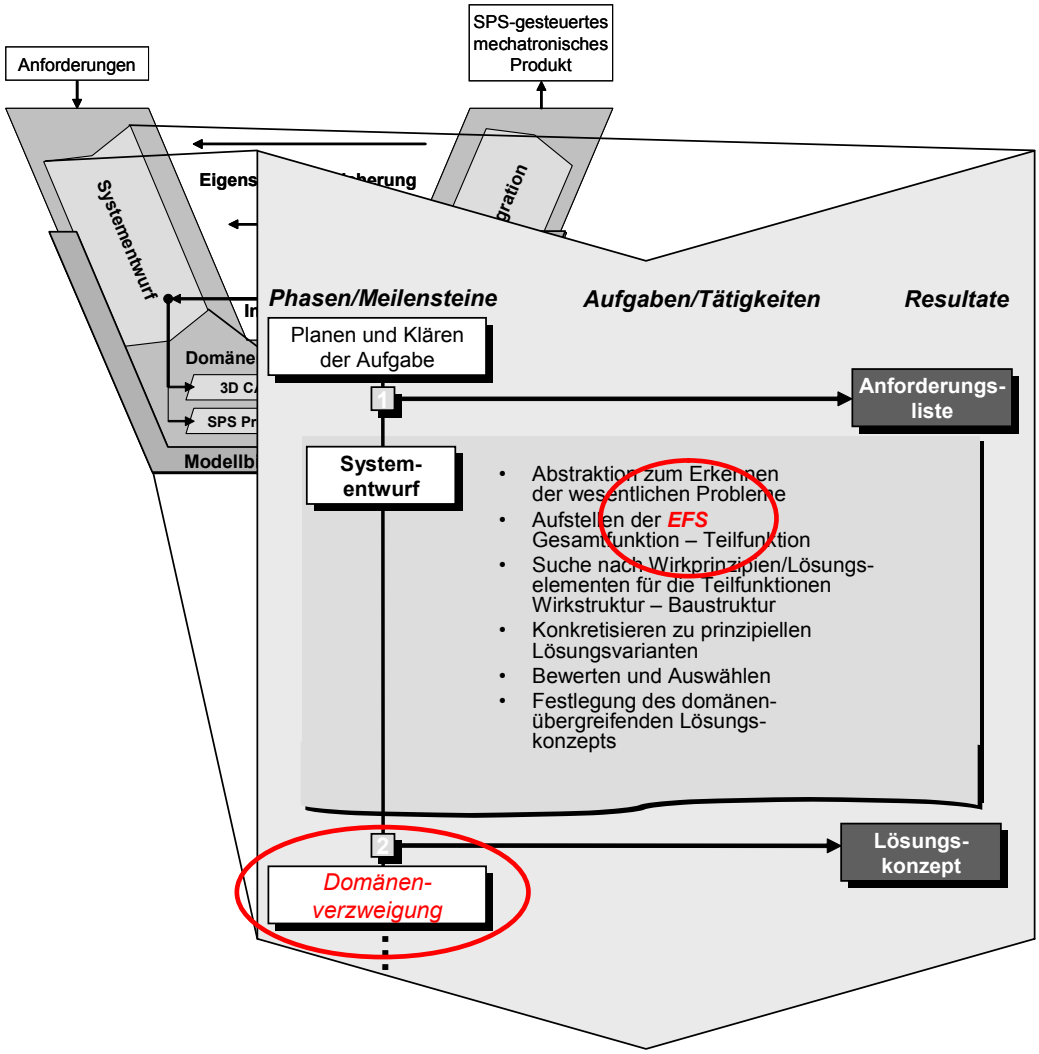


Bild 40: Angepasster vordefinierter Prozessbaustein für den Systementwurf [1]

Wie auch schon in der VDI 2221 ([73], Abschnitt 2.2) für die Konzeptphase in der Konstruktion empfohlen, werden Funktionen eines Systemes definiert, um basierend auf der Anforderungsliste den Lösungsraum abstrakt zu beschreiben. Anschliessend lassen sich z.B. mit dem morphologischen Kasten [59] Teillösungen den Funktionen zuordnen und bewerten. Mit Hilfe der EFS kann nun auch die Ablauflogik für die Steuerungstechnik und die Aktorik/Sensorik integral beschrieben werden. Dadurch lässt sich nicht nur der frühe Einbezug der Steuerungstechnik, sondern auch ein planmässiger Übergang in die nächste Phase sicherstellen (Bild 41). In der Domänenverzweigung werden von dem gemeinsam erstellten Lösungskonzept (Bild 40 und Bild 41) die initialen Daten für beide Domänen im domänenspezifischen Entwurf abgeleitet.

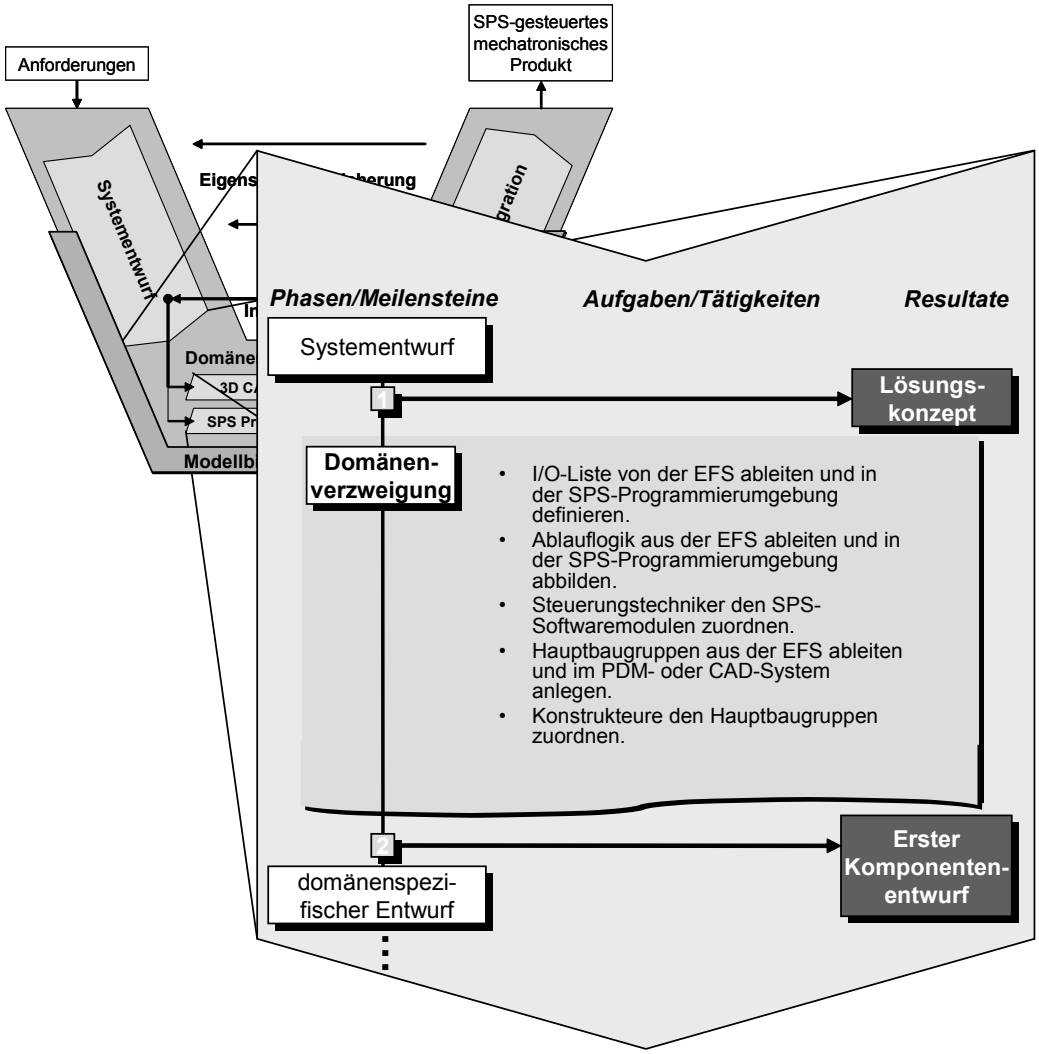


Bild 41: Neuer vordefinierter Prozessbaustein für die Domänenverzweigung [1]

Für den Einstieg in die SPS-Softwareentwicklung und das CAD-Modellieren werden die folgenden Tätigkeiten vorgeschlagen:

- **I/O-Liste von der EFS ableiten und in der SPS Programmierumgebung definieren.**
  Über die Input- und Outputvariablen der I/O-Liste weiss der Steuerungstechniker, welche Aktorik und Sensorik in der Software adressiert werden kann. Diese I/O-Liste wird nun, wie in Abschnitt 3.1.2 beschrieben, von der EFS abgeleitet. Anschliessend müssen die erhaltenen Input- und Outputvariablen in der SPS-Programmierumgebung angelegt werden.
- **Ablauflogik aus der EFS ableiten und in der SPS-Programmierumgebung abbilden.**
  Die Ablauflogik beschreibt das Verhalten des SPS-gesteuerten mechatronischen Systems

aus der Sicht der Steuerung. Durch Vernachlässigung der rein elektromechanischen Funktionen wird diese Ablauflogik aus der EFS abgeleitet (Abschnitt 3.1.2). Anschliessend kann diese z.B. als SFC zusammen mit der I/O-Liste ein erstes kompilierbares Gerüst für die Steuerungstechnik bieten.

- **Steuerungstechniker den SPS-Softwaremodulen zuordnen.**
  Zur Unterstützung des „concurrent engineering“ in der Steuerungstechnik lassen sich nach dem Aufsetzen des Softwareprojektes in der SPS-Programmierumgebung einzelne Softwaremodule den verschiedenen Steuerungstechnikern zuordnen [4].
- **Hauptbaugruppen aus der EFS ableiten und im PDM- oder CAD-System anlegen.**
  Durch die wachsende Integration von Funktionen, Wirkprinzipien/Lösungselementen und Technologien ist das bottom-up-design nicht mehr ausreichend [72]. Zunächst müssen Kenntnisse der Grobstruktur erlangt werden, um diese anschliessend zu verfeinern (top-down-design). Durch das Ableiten der Hauptbaugruppen aus der EFS (Abschnitt 3.1.2) kann diese grobe Struktur im CAD-System, oder falls vorhanden im PDM-System, angelegt werden. Zusätzlich erfährt die Konstruktion die Zuordnung der Aktoren und Sensoren zu den Hauptbaugruppen.
- **Konstrukteure den Hauptbaugruppen zuordnen.**
  Analog zur Steuerungstechnik lässt sich das „concurrent engineering“ innerhalb der Konstruktion unterstützen, indem die abgeleiteten Hauptbaugruppen den verschiedenen Konstrukteuren zugeordnet werden [4].

Somit können nun Konstruktion und Steuerungstechnik gleichzeitig den domänenspezifischen Entwurf beginnen. In dieser Phase wird die Synchronisation beider Domänen wichtig. Änderungen, die sich auf die jeweilig andere Domäne auswirken, sollten umgehend kommuniziert werden [62]. Wie in Abschnitt 3.1.3 beschrieben, leistet die EFS dazu ihren Beitrag um interdisziplinär relevante Änderungen im domänenspezifischen Entwurf (Bild 42) zu identifizieren und gezielt zu verbreiten.

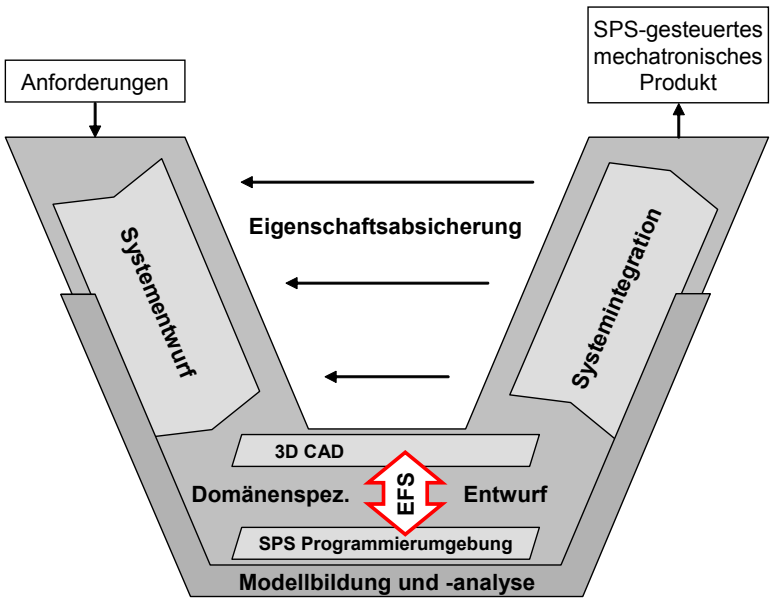


Bild 42: Die EFS als Brücke bei interdisziplinär relevanten Änderungen im domänenspezifischen Entwurf

Dabei spielen die Aktoren und Sensoren im Kontext der EFS (Bild 38) eine zentrale Rolle um festzustellen welche Baugruppen im CAD und welche I/O‘s in der SPS Programmierumgebung jeweils betroffen sind.

Nach dem domänenspezifischen Entwurf ermöglicht die Virtuelle Maschine (Bild 16) eine erste Eigenschaftsabsicherung. Dabei werden die Daten aus dem 3D-CAD mit der Software aus der SPS-Programmierumgebung über eine Eventsimulation gekoppelt [17]. In Bild 43 ist der entsprechende Prozessbaustein für den Einstieg in die Systemintegration dargestellt.

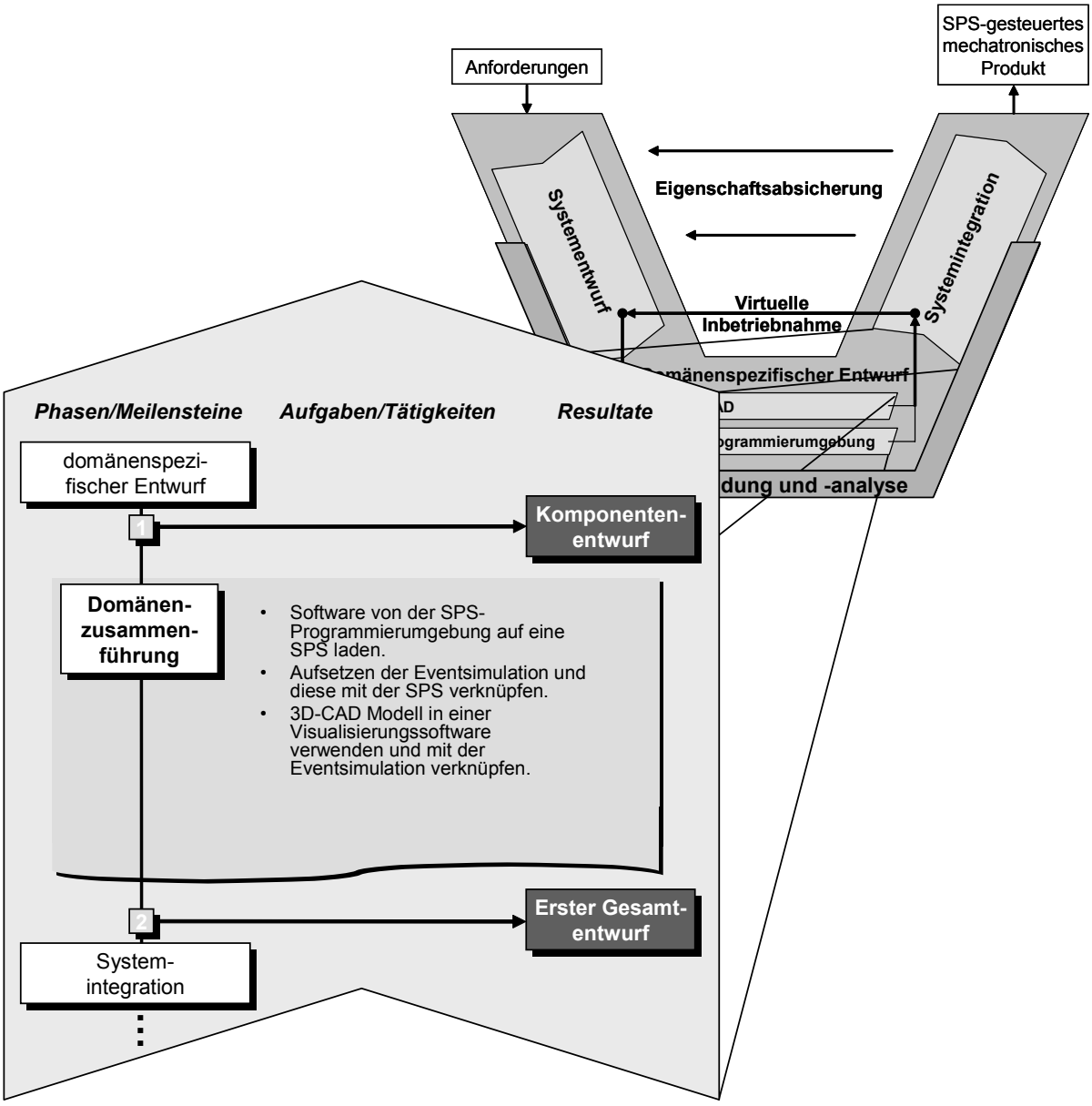


Bild 43: Neuer vordefinierter Prozessbaustein für die Domänenzusammenführung in Anlehnung an [1]

Der erste Gesamtentwurf entspricht hier einer Virtuellen Maschine, mit deren Hilfe eine erste Eigenschaftsabsicherung über eine virtuelle Inbetriebnahme möglich ist. Dazu sind die folgenden Tätigkeiten nötig:

- **Software von der SPS-Programmierumgebung auf eine SPS laden.**
  Die Hardwarekonfiguration der Steuerung, einschliesslich der Kommunikationswege über

die I/O‘s zu den Aktoren/Sensoren, muss in der SPS-Programmierumgebung definiert sein. Ist der Entwurf der Software abgeschlossen, wird diese auf die SPS geladen.

- **Aufsetzen der Eventsimulation und diese mit der SPS verknüpfen.**
  In der Eventsimulation werden die Signale der Steuerung an die Aktoren empfangen und simulierte Signale der Sensoren zurückgeschickt. Dazu muss das Verhalten des mechatronischen Systems in der Eventsimulation abgebildet werden. Zur Verknüpfung der Eventsimulation mit der SPS müssen auch in der Eventsimulation dieselben I/O‘s definiert werden. Wird z.B. die SPS-Programmierumgebung SIMATIC [65] zusammen mit der Eventsimulation SIMIT [66] verwendet, können die I/O‘s direkt von SIMATIC übernommen werden. Ansonsten können die I/O‘s auch, wie in Abschnitt 3.1.2 beschrieben, von der EFS abgeleitet werden.
- **3D-CAD Modell in einer Visualisierungssoftware verwenden und mit der Eventsimulation verknüpfen.**
  Bei grossen 3D-CAD Modellen lohnt es sich, zuerst die Modelle im CAD zu vereinfachen [1]/[17]. So lassen sich z.B. im CAD-System NX von UGS [71] Features wie Rundungen oder Bohrungen und kleine Einzelteile wie Schrauben und Muttern herausfiltern. Danach kann die Geometrie über VRML- oder JT-Dateien vom CAD in eine Visualisierungssoftware (wie z.B. Mediator [33]) importiert werden. Je nach Bedarf wird in der Visualisierungssoftware noch die Kinematik der Maschine und Texturen/Licht/.. definiert. Bei der Verknüpfung mit der Eventsimulation müssen zumindest die Positionswerte der Aktoren an die Visualisierungssoftware übermittelt werden, um die Bewegungen im Zusammenspiel mit der SPS darstellen zu können. Darüber hinaus können Events (z.B. Kollisionen) in der Visualisierungssoftware definiert werden, die dann zurück an die Eventsimulation gesendet werden [1].

Nach der erfolgreichen virtuellen Inbetriebnahme kann nun die Systemintegration, wie in der VDI 2206 [72] beschrieben, fortgesetzt werden.

In diesem Abschnitt und in den VDI-Richtlinien 2221 und 2206 ([73], [72]) wird das planmässige Vorgehen von der Anforderungsliste bis zum Gesamtentwurf des Produktes beschrieben. Diese Vorgehensweise lässt sich nicht nur bei Neukonstruktionen anwenden. Auch Änderungskonstruktionen, bei denen ein bestehendes Produkt zusammen mit neuen Anforderungen ein neues Produkt ergeben soll, lassen sich so durchführen. Die in Abschnitt 3.1.1 vorgestellte EFS

kann hier gut mit der Funktionenanalyse (VDI-Richtlinie 2803 [74]) kombiniert werden. Dazu werden die Ist-Funktionen des bestehenden Produktes analysiert und eine Ist-EFS aufgestellt. Anschliessend wird, basierend auf den neuen Anforderungen und der Ist-EFS, eine Soll-EFS erstellt. Mit dieser Soll-EFS kann nun der Systementwurf weitergeführt werden (Bild 40). Im nächsten Abschnitt wird erläutert, wie das in diesem Abschnitt präsentierte planmässige Vorgehen in Kombination mit den in Abschnitt 3.1 vorgestellten Methoden durch Software unterstützt werden kann.

3.3 Neue Software zur Unterstützung der Methoden

Im Rahmen dieser Arbeit wurde eine Software zur Erfassung und Nutzung der EFS implementiert. Dieses EFS-Tool ist in Bild 44 im Kontext der anderen Entwurfswerkzeuge dargestellt.

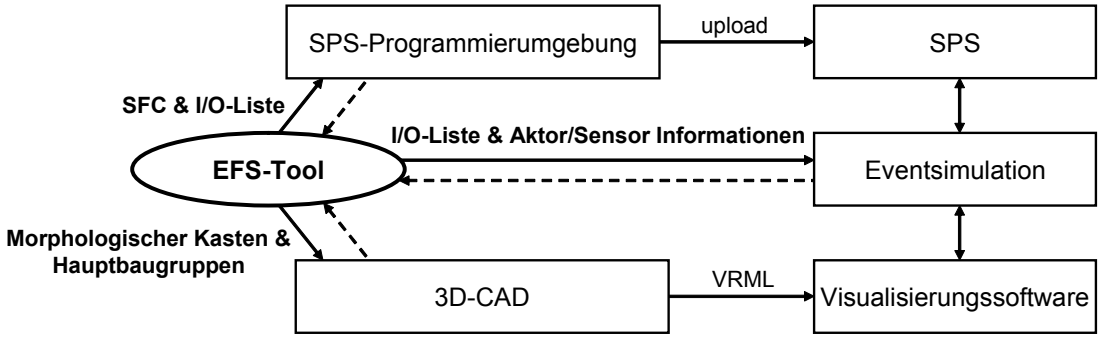


Bild 44: Tools im Kontext SPS-gesteuerter mechatronischer Systeme

Im EFS-Tool lässt sich die EFS, wie in Abschnitt 3.1.1 beschrieben, modellieren. Der Übergang zum domänenspezifischen Entwurf wird unterstützt, indem das SFC und die I/O-Liste für die SPS-Programmierumgebung automatisch aus der EFS generiert wird (Abschnitt 3.1.2). Zur Unterstützung der Lösungsfindung in der Konstruktion kann ein Morphologischer Kasten von der EFS abgeleitet werden. Dieser enthält eine Liste der aktuellen Elementarfunktionen und deren Flüsse. Da die Regeln von Stone heuristisch und nicht algorithmisch sind [67], lassen sich die Hauptbaugruppen nicht automatisch herleiten. Gleichwohl lassen sich die Hauptbaugruppen im EFS-Tool markieren. In Abschnitt 3.1.3 wird aufgezeigt, wie Änderungen im domänenspezifischen Entwurf durch die Aktualisierung der EFS der jeweils anderen Domäne mitgeteilt werden können. Im Beispiel aus Abschnitt 3.1.3, bei dem ein Sensor im CAD entfernt wird,

führt diese Anpassung im EFS-Tool unmittelbar zu einer Löschung der entsprechenden Inputvariablen. Die Verwendung einer aktuellen automatisch abgeleiteten I/O-Liste führt nun zu den beschriebenen Konsistenzmeldungen in der SPS-Programmierumgebung.

Zum Aufsetzen der Eventsimulation werden wiederum die I/O‘s zur Kommunikation mit der SPS benötigt (Bild 43). Zusätzlich müssen die Eigenschaften der Aktoren und Sensoren bekannt sein, um das Modell für die Eventsimulation zu erstellen. Da die verwendeten Aktoren und Sensoren im EFS-Tool erfasst sind, können diese Informationen zusätzlich zu den I/O‘s exportiert werden. Der upload auf die SPS, das Ex- und Importieren der Geometrie und die Verknüpfung der SPS mit der Visualisierungssoftware über die Eventsimulation in Bild 44 ist in [17] und in Bild 43 beschrieben. Somit steht das neu entwickelte EFS-Tool im Kontext der beiden Hauptentwurfswerkzeuge 3D-CAD und SPS-Programmierumgebung (Abschnitt 2.4) und der Tools der Virtuellen Maschine (Bild 16). In Bild 45 sieht man einen Screenshot des EFS-Tools ELVAN.

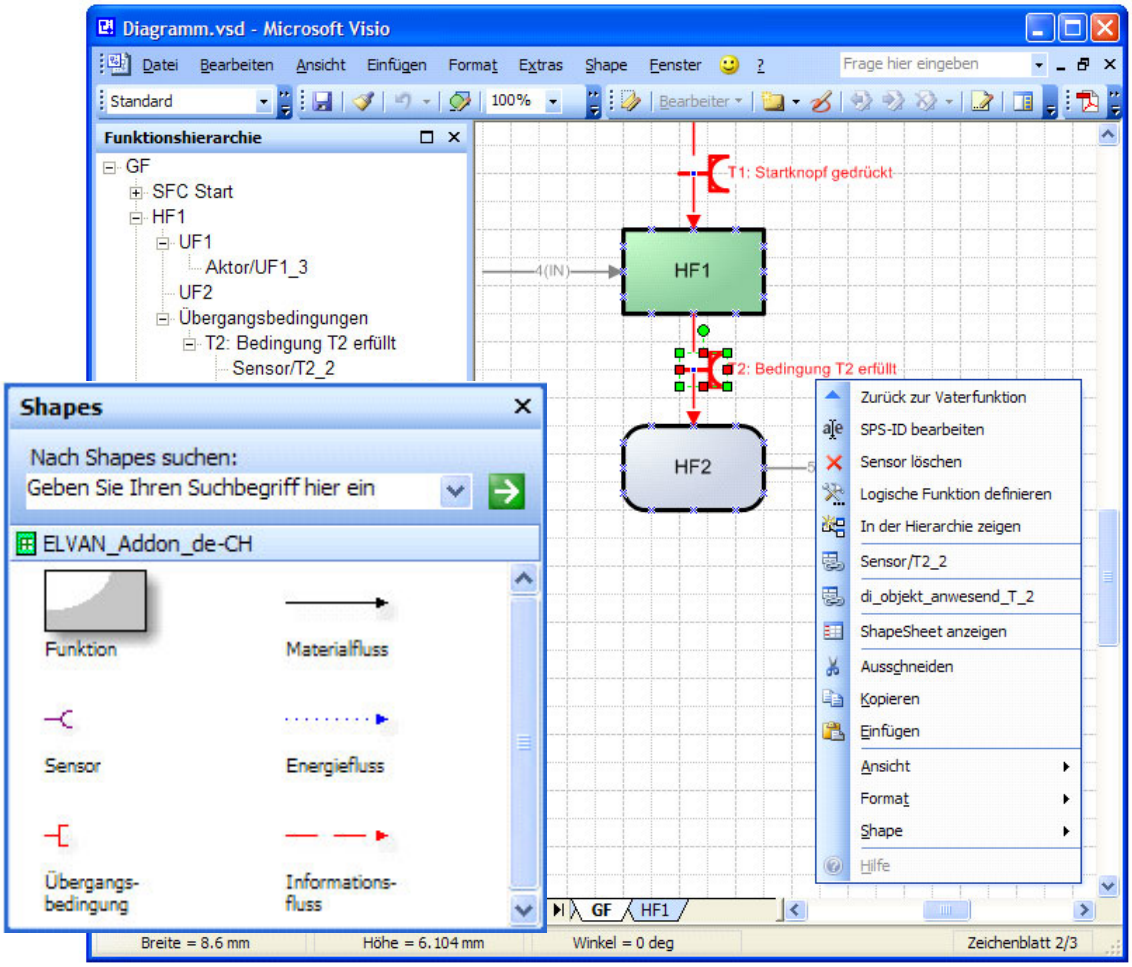


Bild 45: Screenshot von ELVAN

ELVAN erweitert das MS Office Programm Visio [52] um EFS-spezifische Funktionalitäten. Visio [52] eignet sich gut zur Darstellung von Flussdiagrammen. Bei der ablaufbezogenen Darstellung der Funktionsstruktur lassen sich Funktionen und Flüsse dynamisch verbinden. Die Integration von Visio in MS Office und die Erweiterbarkeit hat dazu geführt, Visio als Basistool für die softwaretechnische Unterstützung zu wählen. Vor allem die Erweiterbarkeit durch eigene Softwareentwicklungen war wichtig, um die in Abschnitt 3.1 präsentierten Methoden durchgängig durch Tools zu unterstützen. Diese Erweiterungen sind in C# als Visio Add-On realisiert. Zur konsistenten Verwaltung der Daten wie Aktor-/Sensorinformationen und I/O‘s wird Access im Hintergrund verwendet.

In Bild 45 sind links unten die vordefinierten Master-Shapes in einer eigenen Schablone für Funktionen, Übergangsbedingungen, Material-, Energie- und Informationsflüsse zusammengefasst. Diese Shapes können analog zu den vorinstallierten Visio-Shapes per Drag&Drop in den Zeichnungen verwendet werden. Das Anlegen von Unterfunktionen findet über das Kontextmenü der jeweiligen Funktion statt. Dabei wird ein neues Zeichenblatt (Sheet) erzeugt, in dem neben der neuen Unterfunktion weitere Unterfunktionen definierbar sind.

Bei der Verwendung von Werkzeugen wie Visio zur hierarchischen und ablaufbezogenen Darstellung der Funktionsstruktur müssen jeweils zwei Zeichnungen gemacht werden. Dabei wird eine Funktion redundant in beiden Darstellungen eingetragen. Dies führt neben dem höheren Aufwand vor allem zu Konsistenzproblemen bei Änderungen. Deshalb bestand die erste Erweiterung der Visio Software darin, die Funktionshierarchie automatisch abzuleiten und ständig eine aktuelle Darstellung in einem eigenen Fenster anzuzeigen (Links oben in Bild 45).

Trifft man beim Definieren von Unterfunktionen auf eine Aktorfunktion, kann die Rekursion gestoppt werden (Abschnitt 3.1.1). Steht der Aktor mit seinen Outputvariablen fest, wird dieser, wie in Bild 46 gezeigt, in ELVAN definiert.

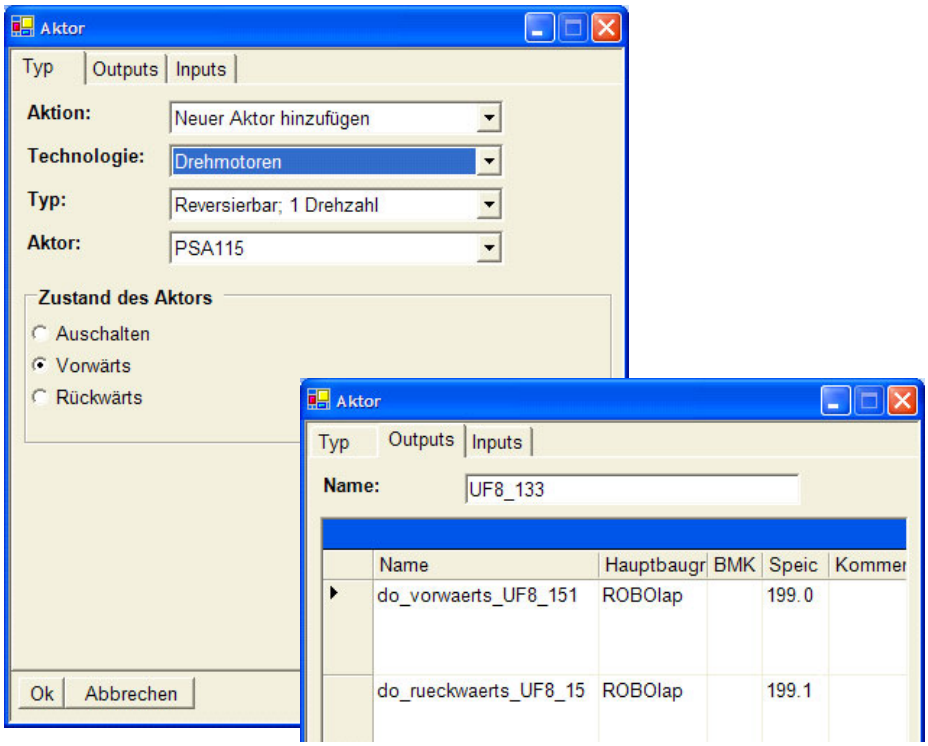


Bild 46: Einen neuen Aktor einer Aktorfunktion zuweisen

Dabei muss zuerst die verwendete Technologie, wie Drehmotoren oder Hydraulikzylinder, gewählt werden. Anschliessend wird der Typ, wie z.B. Bi- oder Monostabiler Zyliner, definiert. Schlussendlich kann man einen spezifischen Aktor aus der Datenbank selektieren. Dazu ist eine kleine Auswahl von Aktoren und Sensoren in ELVAN vordefiniert. Bei der Anwendung von ELVAN empfiehlt es sich im vorhinein firmenspezifische Listen über Access in ELVAN zu definieren (Bild 48). Da zugekaufte Aktoren auch Sensorik beinhalten können, werden neben den Outputvariablen auch die zugehörigen Inputvariablen angelegt. Diese Inputvariablen stehen nun auch bei der Formulierung der Übergangsbedingungen zur Verfügung. Je nach angestrebtem Zustand des Aktors werden automatisch die entsprechenden Outputvariablen für die SPS gesetzt.

Sensoren sind analog zu den Aktoren über die Technologie und den Typ klassifiziert und können entsprechend über das Kontextmenü definiert werden (Bild 57). Wird in einer Übergangsbedingung mehr als nur der Zustand einer Inputvariable abgefragt, können über einen Logikeditor in ELVAN die boolschen Beziehungen der Inputvariablen definiert werden (Bild 47).

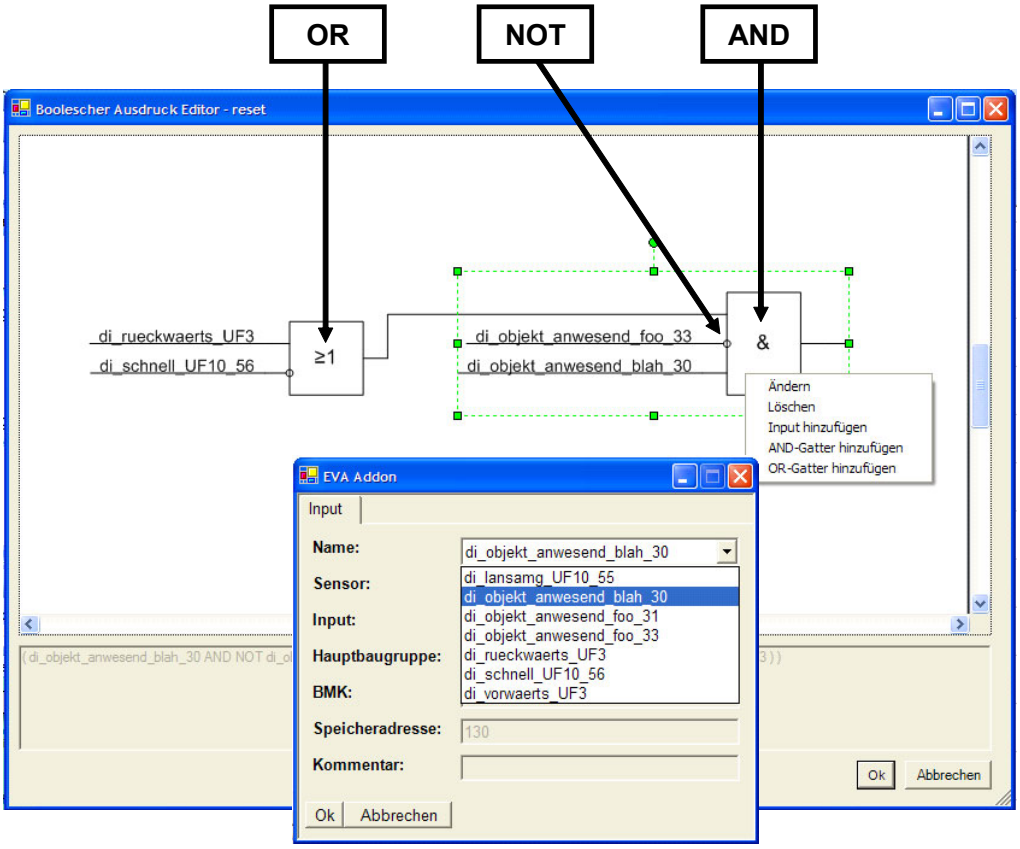


Bild 47: Logikeditor zur Definition von nicht trivialen Übergangsbedingungen

Beliebig viele AND- und OR-Gatter mit jeweils beliebig vielen (negierten) Inputvariablen können hier entsprechend der DIN EN 60617-12 [21] verschachtelt werden.

Die bisher beschriebene Funktionalität in ELVAN erlaubt das effiziente Definieren der EFS basierend auf der in Abschnitt 3.1.1 vorgestellten Methode. Dazu sollten Aktoren und Sensoren, die typischerweise in einer Firma Verwendung finden, schon im Vorfeld angelegt werden. Solche firmenspezifischen Anpassungen können von einem Administrator in Access durchgeführt werden (Bild 48).

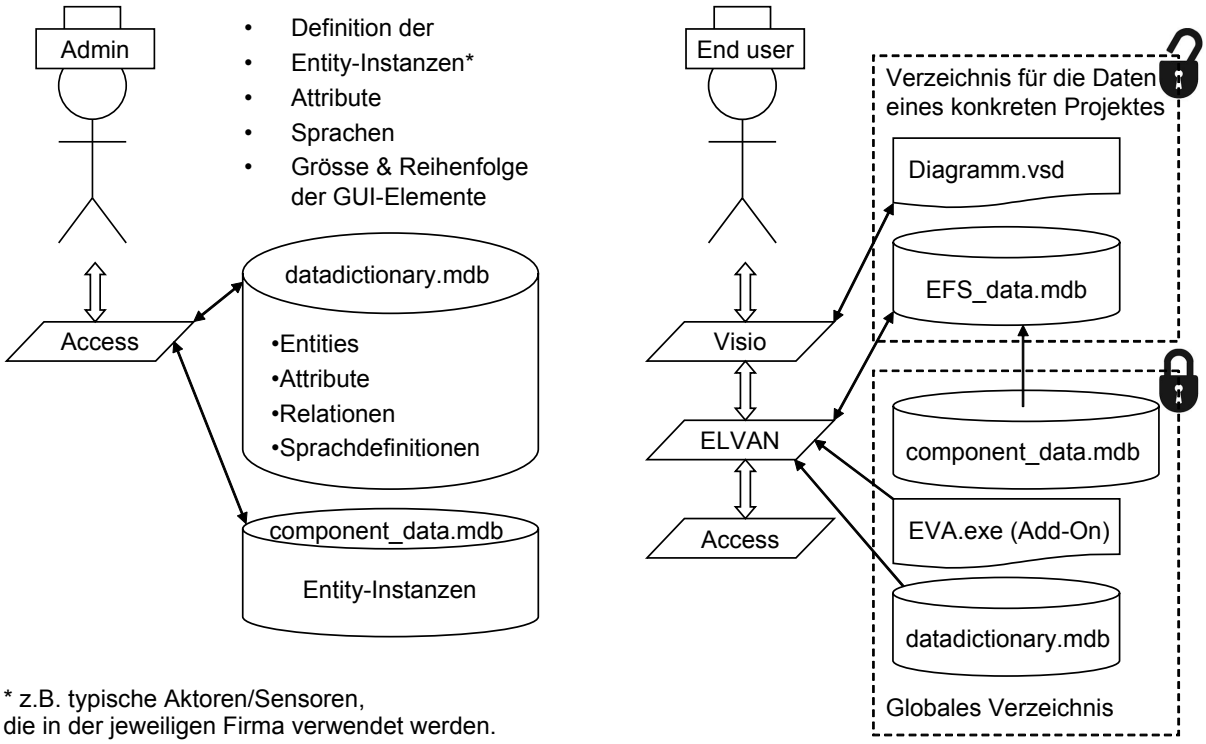


Bild 48: Firmenspezifische Anpassung von ELVAN

Dabei werden neben den vordefinierten Aktoren und Sensoren (in „component_data.mdb“) auch Sprachversionen und GUI-Elemente (in „datadictionary.mdb“) über Access verwaltet. Diese Dateien dienen nun als schreibgeschützte Datenbank allen Benutzern innerhalb einer Firma. Wird nun z.B. ein Aktor über die EFS in Visio ausgewählt, dann instanziert ELVAN diesen Aktor und speichert ihn in der Datei „EFS_data.mdb“.

Aus Bild 44 ist ersichtlich, dass das EFS-Tool ELVAN nicht nur zur Modellierung der EFS Anwendung findet. Auch Tools der nachgelagerten Phasen im Entwicklungsprozess sind angebunden. So lässt sich aus der EFS automatisch, basierend auf der Methode aus Abschnitt 3.1.2, die I/O-Liste und das SFC ableiten. In Bild 49 ist ein Beispiel für die SPS-Programmierumgebung SIMATIC [65] von SIEMENS dargestellt.

```
126,diPush2EmptyCanFrontPos I 0.0 BOOL
126,diPush2EmptyCanBackPos  I 0.1 BOOL
126,diPushEmptyCanFrontPos  I 0.2 BOOL
126,diPushEmptyCanBackPos   I 0.3 BOOL
126,diCentRollClosed        I 0.4 BOOL
126,diCentRollOpen          I 0.5 BOOL
126,diPushFullCanFrontPos   I 0.6 BOOL
126,diPushFullCanBackPos    I 0.7 BOOL
126,di_ein_Hauptmotor_st_12 I 1.0 BOOL
126,di_ein_Weitleuchte_e_13 I 1.1 BOOL
126,diStripTransportOk      I 1.2 BOOL
126,di_grundstellung_Ban_15 I 1.3 BOOL
126,diNotSkStop             I 1.4 BOOL
126,diButStart              I 1.5 BOOL
126,diButNotStop            I 1.6 BOOL
126,diLapEmpty              I 1.7 BOOL
:
126,doPush2EmptyCanFrwd     Q 0.0 BOOL
126,doPushEmptyCanFrwd      Q 0.1 BOOL
126,doCentRollOpening       Q 0.2 BOOL
126,doPushFullCanFrwd       Q 0.3 BOOL
:
```

I/O-Liste

```
:
STEP V2_Kannen_schieber_ (*$_NUM 16*):
(*$_COM V2_Kannen_schieber_einfa@Leerk
annen- schieber einfahren*)
"doPush2EmptyCanFrwd"   (R)
END_STEP
:
TRANSITION Sicherheitskrei (*$_NUM 7*)
  FROM Unterstuetzung_Bandtrans
  TO Hauptmotor_stoppen
CONDITION := ( NOT "diButNotStop" OR
NOT "diNotSkStop" OR "diLapEmpty" OR
"diCanFull" OR NOT
"diCanNotOverfilled" )
END_TRANSITION
:
```

SFC

Bild 49: I/O-Liste und SFC im SIMATIC-Format nach dem Export aus ELVAN

In dieser I/O-Liste sind alle Input und Outputvariablen aufgelistet. Jede Variable ist als Input (I) oder Output (Q) markiert. Eine konsistente Defaultadressierung wird in ELVAN automatisch erzeugt. Dabei entspricht der erste Wert dem Byte und der zweite dem Bit. D.h. ein I/O auf der Adresse 0.2 referenziert auf das erste Byte und innerhalb dieses Bytes auf das dritte Bit. Schlussendlich muss auch der Typ, z.B. BOOL, in dieser I/O-Liste definiert sein.

In der ASCII-Datei der Firma SIEMENS für das SFC werden zuerst die „Steps“ aufgelistet. In diesen Steps werden Outputvariablen für die Aktoren gesetzt. Dabei steht „(R)“ für „Reset“ und führt dazu dass die entsprechende Outputvariable den Wert „0“ (FALSE) zugewiesen bekommt. Den Wert „1“ (TRUE) nimmt die Outputvariable bei einem „(S)“ an was stellvertretend für „Set“ steht. Am Ende der Datei sind dann die „Transitionconditions“ aufgeführt. Sie enthalten die logisch verknüpften Inputvariablen aus den Übergangsbedingungen in ELVAN. Nach dem Import dieser beiden Dateien präsentiert sich das SFC in SIMATIC wie in Bild 50 dargestellt.

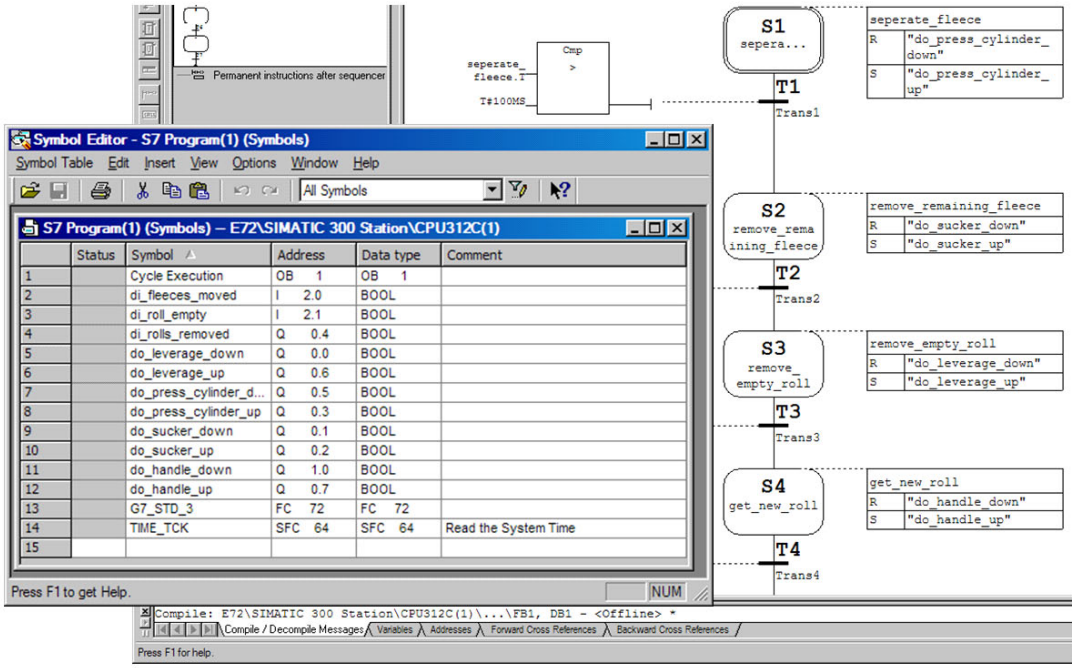


Bild 50: I/O-Liste und SFC nach dem Import in SIMATIC

Das SFC wird nun kompiliert und bei Bedarf erweitert. Zusätzlich lassen sich auch die vier anderen SPS-Programmiersprachen in Kombination mit dem SFC verwenden (Abschnitt 2.3). Durch das Laden des Kompilates auf die integrierte Soft-SPS PLCSIM, kann jederzeit ein erster Test der Software durchgeführt werden. In dieser Soft-SPS lassen sich per Mausklick Inputvariablen ändern und der aktuelle Step im SFC anzeigen. Dadurch lässt sich die Ablauflogik Schritt für Schritt visualisieren. Ist das SPS-Progamm fertig, wird es auf die SPS des mechatronischen Systems geladen. Damit ist der, in Bild 44 skizzierte, Weg von der EFS über das SFC bis zur SPS beschrieben.

Zur Unterstützung der Lösungsfindung in der Konstruktion ist es in ELVAN möglich, einen Morphologischen Kasten (Abschnitt 2.2) von der EFS als Exceltabelle wie in Bild 51 dargestellt abzuleiten. Im Morphologischen Kasten werden zu gegebenen Funktionen verschiedene Teillösungen gesucht. Dies geschieht mit Vorteil im Team. Die Teillösungen werden bewertet und zu einem funktionsübergreifenden Lösungskonzept vereint (Bild 40 und Bild 61).

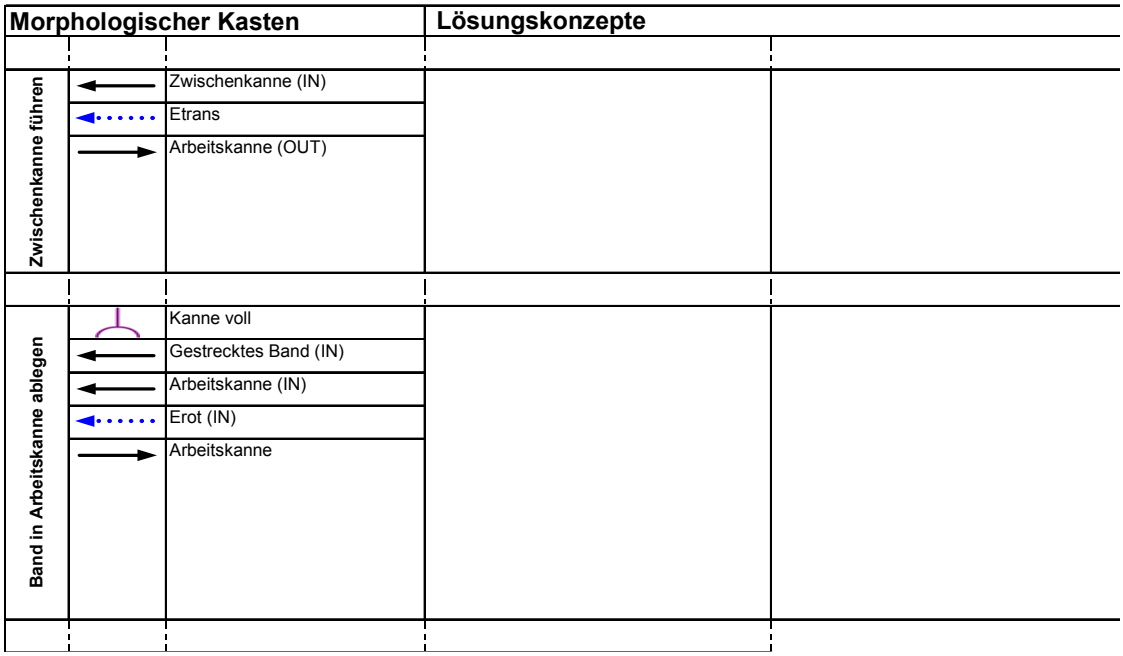


Bild 51: Ausschnitt aus einem von ELVAN generierten Morphologischen Kasten in Excel

Wird der Morphologische Kasten von der EFS abgeleitet, werden zusätzlich zum Funktionsnamen die ein- und ausgehenden Flüsse eingetragen. Darüber hinaus werden auch die zu den Funktionen assoziierten Sensoren neben den Flüssen exportiert um die zu diskutierende Funktion in einen Kontext zu stellen. Aktorfunktionen mit spezifiziertem Aktor werden nicht mehr aufgelistet, da hier die Umsetzung der Funktion schon definiert ist. Zusätzlich werden Funktionen mit Unterfunktionen gefiltert, da hier auch schon eine Konkretisierung stattgefunden hat. Somit verbleiben alle Elementarfunktionen welche keine Aktorfunktionen sind in der Exceltabelle (Bild 51) nach dem Export aus ELVAN.

Bei der virtuellen Inbetriebnahme durch die Virtuelle Maschine (Bild 16) wird mit der Eventsimulation die Brücke zwischen der SPS der Steuerungstechnik und dem 3D-Modell der Konstruktion geschlagen. Zur effizienten Erstellung dieser Eventsimulation lässt sich eine erweiterte I/O-Liste von der EFS ableiten (Bild 52).

Nr.	1	2	7
_Symbolname Operanden	diCentRollClosed	diNotSkStop	doCentRollOpening
Adresse im WinMOD	XE0.5	XE3.4	XA8.0
Typ	BI	BI	BO
Defaultwert	0	0	0
Kommentar	Positioniert Kanne		Positionierung Kanne
Produktmodul	Kannenwechsel	Test Sicherheitskreis	Kannenwechsel
Simulationsmodul	Zentrierrolle		Zentrierrolle
Stellzeit/Geschwindigkeit/Weg			3s
Betriebsmittelkennzeichen	B58	K05	Y01
Herstellerbezeichnung der Aktoren und Sensoren			Festo - DGS-25

Bild 52: Erweiterte I/O-Liste zur automatischen Kopplung der Eventsimulation WinMOD [50] *mit einer SPS und zusätzlichen Aktor- und Sensorinformationen zur Unterstützung der Modellerstellung einer Eventsimulation*

In der erweiterten I/O-Liste sind analog zur I/O-Liste für SIMATIC (Bild 49) die notwendigen Informationen zu den I/O‘s abgelegt. In Bild 52 ist ein Beispiel für die Eventsimulation WinMOD [50] dargestellt. Eine Exceltabelle in diesem Format erlaubt das automatische Importieren der Schnittstellenvariablen zur SPS in WinMOD. Darüber hinaus können zusätzliche Kommentare in WinMOD importiert werden. So z.B. Informationen über die Aktortechnologie und den Aktortyp, wie sie schon in ELVAN bei der Definiton der Aktoren (Bild 46) hinterlegt sind. Auch Aktor-Parameter, wie die Stellzeit eines Pneumatikzylinders, können in ELVAN verwaltet werden. In Abhängigkeit dieser zusätzlichen Informationen kann nun in WinMOD das passende Aktorelement (Makro) ausgewählt, parametrisiert und mit den entsprechenden I/O‘s verknüpft werden. Weitere Informationen, wie das Modul, indem dieser Aktor verwendet wird, helfen den Aktor in einen Kontext zu stellen. Analoges gilt für die Sensoren. Mit Hilfe dieser erweiterten I/O-Liste kann durch den direkten Import der I/O‘s und der gezielten Bereitstellung von ergänzenden Informationen zu der Aktorik und Sensorik der Aufwand zur Erstellung der Eventsimulation reduziert werden.

Insgesamt ergeben sich für das EFS-Tool ELVAN im Kontext der anderen Tools (Bild 44) die folgenden Schlüsselfunktionalitäten:

- Vordefinierte Master-Shapes in einer eigenen Schablone für Funktionen, Übergangsbedingungen, Material-, Energie- und Informationsflüsse
- Assoziative Funktionshierarchie

- Konsistente Verwaltung der Funktionen, Aktoren, Sensoren, Übergangsbedingungen, Outputvariablen und Inputvariablen
- Logikeditor zur Definition von nicht trivialen Übergangsbedingungen
- Export des SFC und der dazugehörigen I/O-Liste
- Export des Morphologischen Kastens
- Export der erweiterten I/O-Liste für WinMOD

Die hier aufgelisteten Funktionalitäten sind nicht im Lieferumfang von Visio enthalten und zeigen die Erweiterungen Visios durch ELVAN auf. Der praktische Einsatz des EFS-Tools ELVAN in Verbindungen mit den anderen Tools aus Bild 44 wird anhand eines Beispieles aus der Industrie im nächsten Kapitel diskutiert.

4 Fallstudie

In diesem Kapitel wird die Entwicklungsmethodik aus Kapitel 3 exemplarisch auf eine Kämmmaschine der Firma Rieter [61] angewandt. Bild 44 zeigt die dazu notwendigen Tools auf. Die in dieser Fallstudie konkret eingesetzte Software ist in Bild 53 eingetragen.

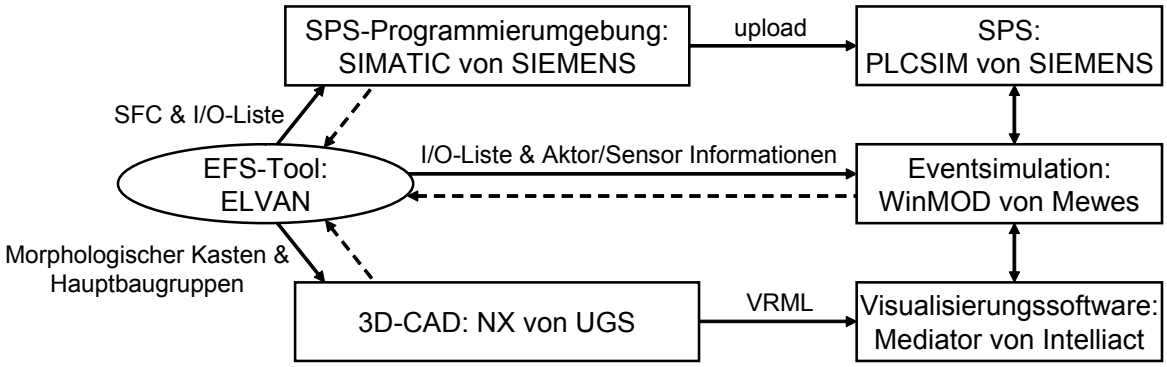


Bild 53: Verwendete Software zur beispielhaften Anwendung der neuen Entwicklungsmethodik auf eine Kämmmaschine

Neben dem neu entwickelten EFS-Tool ELVAN (Abschnitt 3.3) kam das 3D-CAD NX von UGS [71] und die SPS-Programmierumgebung SIMATIC von SIEMENS [65] zum Einsatz. Zur Eigenschaftsabsicherung mittels einer virtuellen Inbetriebnahme (Bild 43) der Kämmmaschine wurde WinMOD [50] mit der Soft-SPS PLCSIM [65] und Mediator [33] kombiniert.

Bevor in Abschnitt 4.2 und Abschnitt 4.3 die Erstellung und Nutzung der EFS für die Kämmmaschine beschrieben wird, erläutert der folgende Abschnitt die Funktion der Kämmmaschine im Kontext des Spinnprozesses.

4.1 Die Kämmmaschine der Firma Rieter im Spinnprozess

Die Firma Rieter versteht sich als Systemanbieter für den gesamten Spinnprozess von der Stapelfaser bis zum Garn. Stapelfasern (auch „Spinnfasern“ oder im englischen „staple fibre“) sind Textilfasern mit begrenzter Länge [20], [35]. Wolle und Baumwolle enthalten z.B. Stapelfasern. Im Gegensatz dazu stellen Endlosfasern (auch „Filamente“ oder im englischen „filament“) Tex-

tilfasern praktisch unbegrenzter Länger dar [20], [35]. So können Chemiefasern wie Polyester und Elasthan theoretisch endlos lang ausgesponnen werden.

Es existieren hauptsächlich zwei Spinnverfahren: das Ring- und das Rotorspinnen. Das Rotorspinnen umfasst weniger Prozessschritte und zeichnet sich durch eine höhere Produktivität aus. Es eignet sich zur Herstellung gröberer Garne die weniger reissfest sind. Da die Kämmmaschine im Rotorspinnprozess nicht zur Anwendung kommt, wird im Folgenden der Ringspinnprozess vertieft.

Um hochwertige Garne zu erhalten sind im Ringspinnprozess die folgenden Schritte nötig [61]:

1. Stapelfaserballen putzen
2. Flocken kardieren
3. Bänder strecken
4. Band wickeln
5. Vliese kämmen
6. Bänder strecken
7. Band vorspinnen
8. Vorgarn ringspinnen

Das Rohmaterial wie z.B. Baumwolle oder Wolle wird in Ballenform angeliefert. Diese Stapelfaserballen werden in einem ersten Schritt geputzt. Dabei werden die Fasern im Ballen zuerst vom Ballenöffner grob voneinander gelöst. Anschliessend werden die Rohfasern von Schmutz und Fettresten bei der Wolle oder von Resten der Samenkapseln bei der Baumwolle befreit. Die entstehenden Flocken werden gemischt. Einen ersten Schritt zum parallelisieren der Fasern stellt das Kadieren dar. Durch ein Harken der losen Fasern in den Flocken ensteht ein Band. Zur Erhöhung der Gleichmässigkeit oder zum Herstellen von Mischfasern werden mehrere Bänder zu einem Band verstreckt. Das Kämmen ist ein optionaler Schritt im Ringspinnprozesss um qualitativ hochwertige und feine Garne zu erhalten, indem durch Ausscheidung von Kurzfasern die durchschnittliche Faserlänge erhöht wird. Dazu werden vorbereitend Vlieswickel produziert. Auch wenn das optionale Kämmen nicht stattfindet, wird das Band ein zweites Mal gestreckt, bevor es zu einem Vorgarn versponnen wird. Abschliessend wird beim Ringspinnen das Vorgarn um den Faktor 40-50 gestreckt und gleichzeitig verdreht.

Die für diese Fallstudie ausgewählte vollautomatisierte Kämmmaschine der Firma Rieter ist in Bild 54 dargestellt. Der Kämmvorgang wird lediglich durch die periodische Auswechslung der Wickel und Kannen unterbrochen.

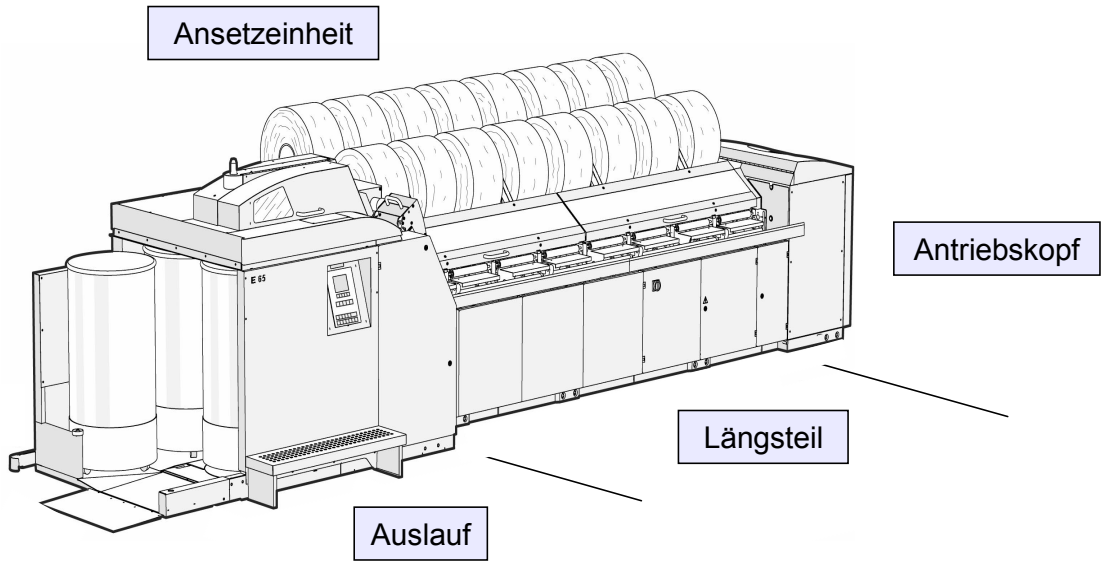


Bild 54: Kämmmaschine mit Hauptbaugruppen

Die Kämmmaschine besteht aus den folgenden Hauptbaugruppen:

- Ansetzeinheit
- Antriebskopf
- Längsteil
- Auslauf

In der Ansetzeinheit werden die Wickel aufgenommen und das Vlies abgewickelt. Über den Antriebskopf wird die mechanische Energie für den Kämmprozess bereitgestellt. Im Längsteil findet das Kämmen statt und im Auslauf werden die Kannen mit dem resultierenden Band gefüllt.

Zur Steuerung der Kämmmaschine kommt die SPS Power Panel PP41 von B&R [7] zum Einsatz. Der Informationsaustausch mit den Aktoren und Sensoren findet über den CAN-Bus (Controller Area Network, [34]) statt. Zur Veranschaulichung der in Kapitel 3 vorgestellten

Methodik wird in den folgenden zwei Abschnitten die EFS für den Prozessschritt des Kämmens erstellt und anschliessend genutzt.

4.2 Erstellung der EFS in ELVAN

Bei der Erstellung der EFS (Abschnitt 3.1.1) wird zuerst die Gesamtfunktion definiert. Im vorliegenden Beispiel der Kämmmaschine lautet sie „Vlies kämmen“ (Bild 55).

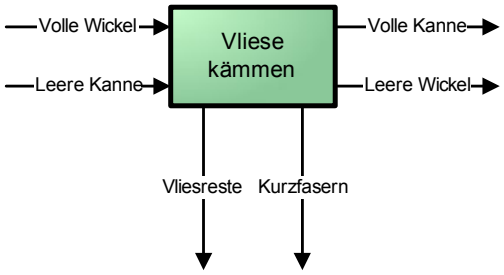


Bild 55: Gesamtfunktion der Kämmmaschine

Die Materialflüsse sind durch den Ringspinnprozess gegeben. Volle Wickel mit dem Vlies werden verarbeitet und das resultierende Band verlässt in der gefüllten Kanne die Maschine. Dazu müssen leere Kannen zugeführt und leere Wickel abgeführt werden. Der Prozess selber - bei dem die Kurzfasern zur Qualitätssteigerung aus dem Vlies herausgekämmt werden - führt dazu, dass diese Kurzfasern abgeführt werden müssen. Auch die Vliesreste, die beim Abwickeln entstehen, müssen aus der Maschine entfernt werden.

Im EFS-Tool ELVAN sind die Hauptfunktionen der Kämmmaschine, wie in Bild 56 dargestellt, modelliert.

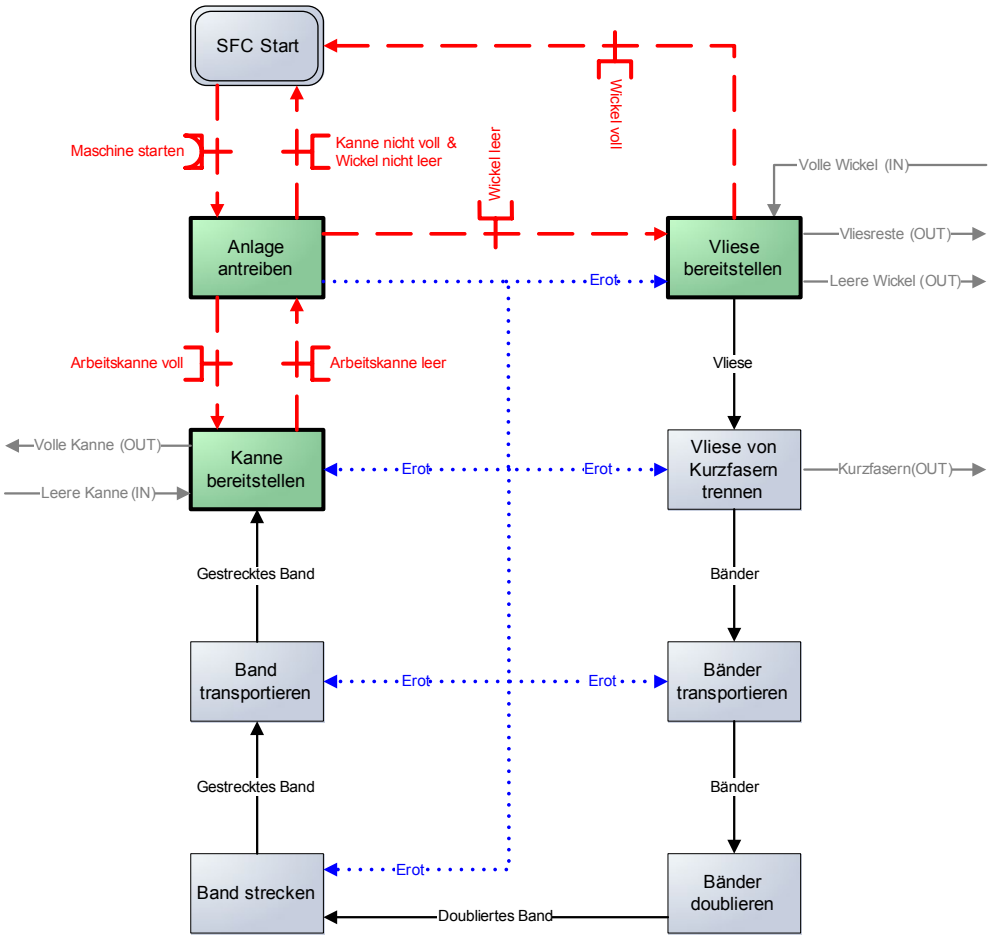


Bild 56: Die Hauptfunktionen der Kämmmaschine in ELVAN

Hier bietet es sich an, zuerst die Hauptfunktionen zusammen mit dem Materialfluss zu modellieren. Aus den vollen Wickeln muss das Vlies bereitgestellt werden. Dabei fallen Vliesreste und leere Wickel an. Aus dem bereitgestellten Vlies werden nun die qualitätsvermindernden Kurzfasern aus dem Vlies entfernt. Die entstehenden Bänder werden transportiert und doubliert, wodurch das doublierte Band entsteht. Dieses Band wird gestreckt und in einer Kanne abgelegt. Die Funktion „Kanne bereitstellen“ ist für den Austausch der gefüllten Kannen mit neuen leeren Kannen verantwortlich.

Die für den Kämmprozess nötige mechanische Energie wird über die Funktion „Anlage antreiben“ bereitgestellt. Diese Funktion erhält die Information zum Starten der Maschine, falls die Übergangsbedingung „Maschine starten“ erfüllt ist. Dieser Informationsfluss ist mit der speziellen Funktion „SFC Start“ verknüpft, welche den Ausgangspunkt des SPS-Programmes kenn-

zeichnet. Falls die Wickel leer sind, wird über den entsprechenden Informationsfluss ein Wickelwechsel in der Funktion „Vliese bereitstellen“ angestossen. Analoges gilt für den Wechsel der Kannen, sobald die Übergangsbedingung „Arbeitskanne voll“ erfüllt ist.

Wie in Abschnitt 3.1.1 erwähnt, lassen sich den Übergangsbedingungen Sensoren zuordnen. In Bild 57 ist die Hinterlegung eines Startknopfes für die Übergangsbedingung „Maschine starten“ dargestellt.

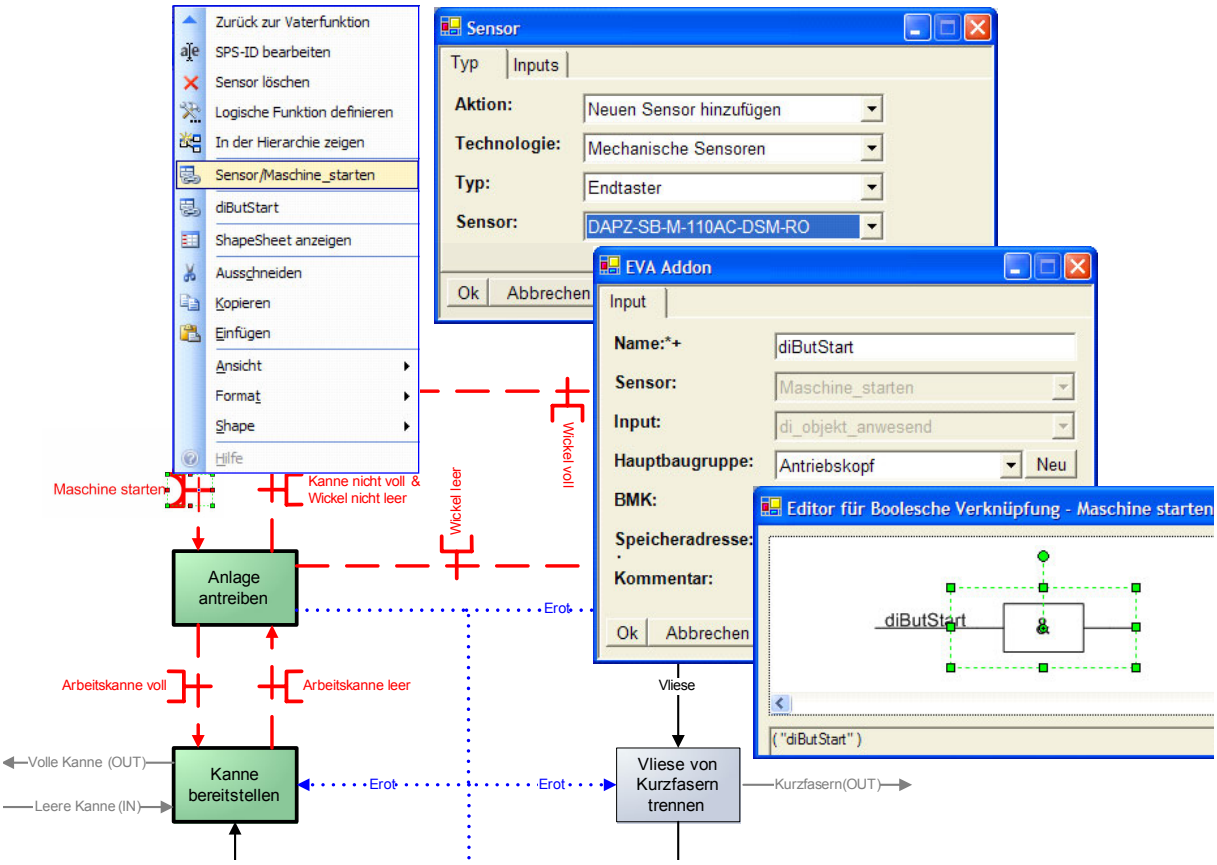


Bild 57: Definition eines Sensors der Kämmmaschine in ELVAN

Bei der Auswahl des mechanischen Endtasters wurde die automatisch generierte Inputvariable zu „diButStart“ geändert. Unmittelbar nach der Definition des Sensors erscheint ein graphischer Editor, in dem die in Bild 57 rechts unten gezeigte Logik vorgeschlagen wird. Das Negieren oder eine logische Verknüpfung mit weiteren Inputvariablen ist wie in Bild 47 gezeigt möglich. In diesem Beispiel des Startknopfes reicht es aus, die voreingestellte Logik zu quittieren.

Nach der Definition der Hauptfunktionen werden diese sukzessive konkretisiert. So enthält die Funktion „Kanne bereitstellen“ unter anderem die Unterfunktion „Zwischenkanne nachrükken“, welche wiederum die in Bild 58 gezeigte Unterfunktionen enthält.

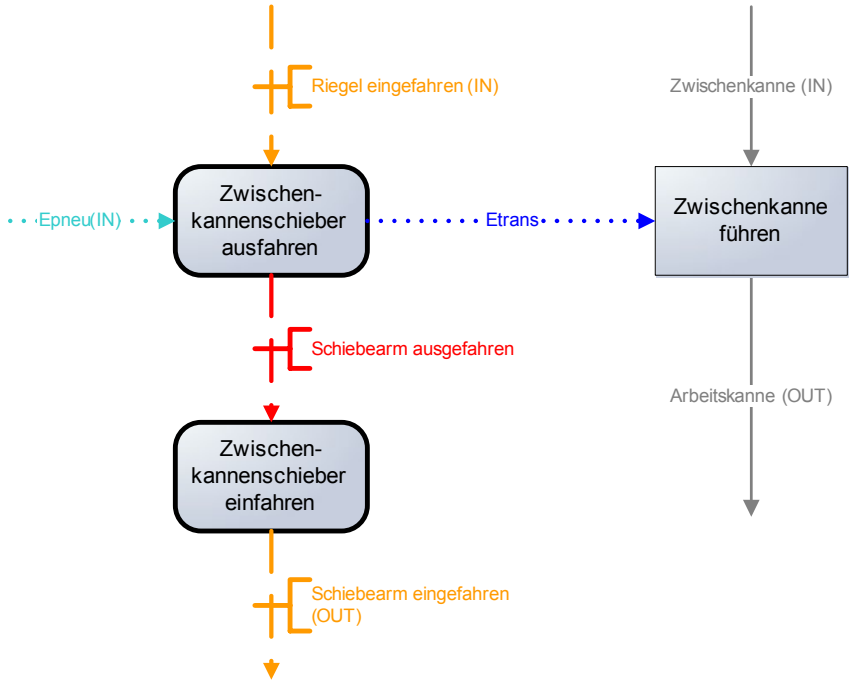


Bild 58: Unterfunktion “Zwischenkanne nachrücken”

In der Funktion „Zwischenkanne nachrücken“ wird die leere Zwischenkanne auf die Arbeitsposition geschoben, wodurch aus dieser Kanne die zu füllende Arbeitskanne wird. Die Ein- und Ausgangsflüsse dieser Funktion, wie z.B. „Zwischenkanne (IN)“ und „Arbeitskanne (OUT)“, sind auf dieser Hierarchieebene lediglich referenziert. Definiert wurden sie schon mindestens eine Hierarchiestufe höher. ELVAN legt diese Flussreferenzen automatisch beim Definieren einer Unterfunktion an. Somit ist die Konsistenz sämtlicher Flüsse über alle Hierarchiestufen hinweg sichergestellt. Auf dieser Hierarchieebene ist den zwei Funktionen mit Informationsfluss jeweils eine Aktorinformation hinterlegt. In Bild 59 (I) ist dargestellt, wie der Funktion „Zwischenkannenschieber ausfahren“ der monostabile Pneumatikzylinder „DGS-25“ mit der Anweisung auf Arbeitsstellung zu fahren („Einschalten“) assoziiert wurde. In der Funktion „Zwischenkannenschieber einfahren“ ist definiert, dass dieser Pneumatikzylinder zurück auf die Grundstellung („Ausschalten“) fahren soll. Dadurch wird die Outputvariable „doPush-

EmptyCanFrwd“ automatisch bei der ersten Funktion auf „1“ (TRUE) und bei der zweiten Funktion auf „0“ (FALSE) gesetzt (II in Bild 59).

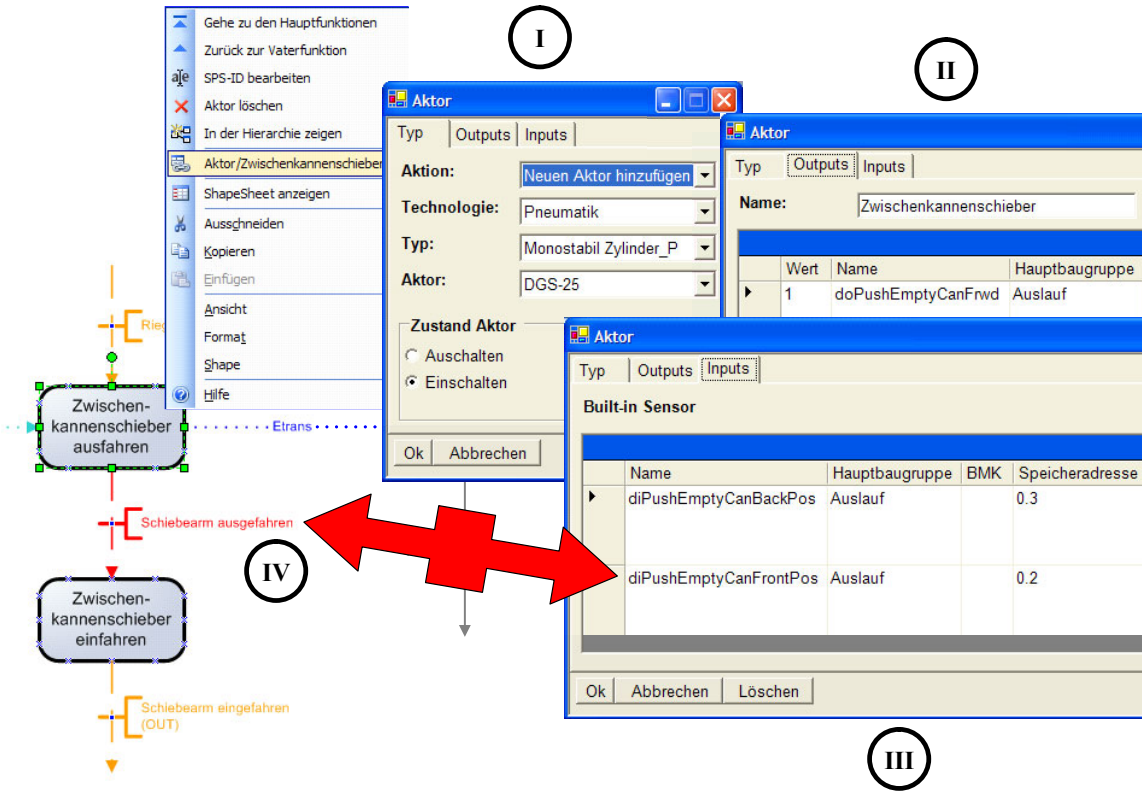


Bild 59: Definition eines Aktors der Kämmmaschine in ELVAN

Der gewählte Pneumatikzylinder „DGS-25“ enthält zwei Sensoren zur Detektion der Arbeits- und Grundstellung des Zylinders. Der zweiwertige Status der Sensoren kann über die SPS-Adressen 0.2 und 0.3 oder über die Inputvariablen „diPushEmptyCanBackPos“ und „diPush-EmptyCanFrontPos“ im SPS-Programm abgefragt werden (III in Bild 59). Diese Inputvariablen stehen nun im gesamten ELVAN-Projekt zur logischen Verknüpfung in Übergangsbedingungen zur Verfügung. In der Übergangsbedingung „Schiebearm ausgefahren“ wurde nach der Definition des Pneumatikzylinders „DGS-25“ dessen zweite Inputvariable „diPushEmptyCan-FrontPos“ ohne zusätzliche boolesche Operationen verwendet (IV in Bild 59). Dadurch wird der Zwischenkannenschieber unmittelbar nach Erreichen der Arbeitsstellung wieder eingefahren.

Die Unterfunktionen in Bild 58 veranschaulichen die zunehmende Konkretisierung der EFS bei zunehmender Hierarchiestufe von der Gesamtfunktion bis zur Aktorfunktion. ELVAN aktualisiert beim Definieren von Funktionen automatisch die in Bild 60 gezeigte hierarchische Darstellung der EFS.

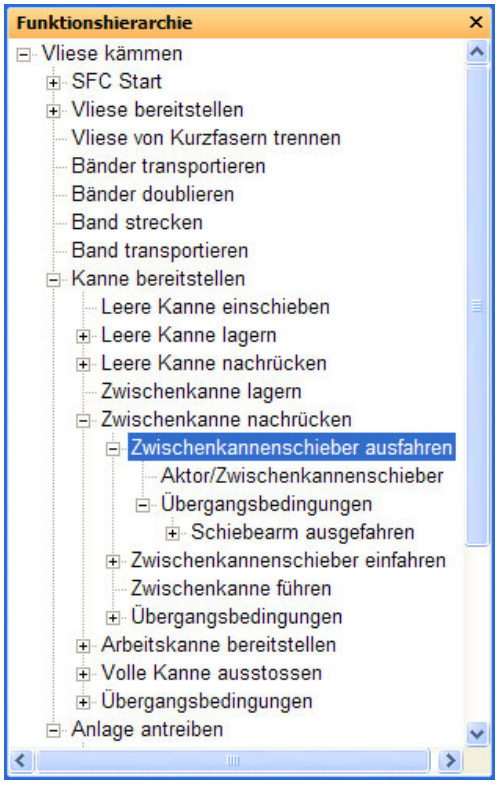


Bild 60: Hierarchische Darstellung der EFS für die Kämmmaschine

Über das Kontextmenu der Funktion „Zwischenkannenschieber ausfahren“ (Bild 59) lässt sich diese Funktion gezielt über die Hierarchieansicht erreichen (Bild 60). Unterhalb der Aktorfunktion kann auf die Aktordefinition und die nachfolgenden Übergangsbedingungen direkt zugegriffen werden. Nachdem in diesem Abschnitt die Erstellung der EFS für die Kämmmaschine gezeigt wurde, beschreibt der Folgende wie ELVAN die nachfolgende Nutzung der EFS unterstützt.

4.3 Nutzung der in ELVAN definierten EFS

ELVAN bietet nach der Definition der EFS Unterstützung zum Starten des Konstruktionsentwurfs, des Steuerungsentwurfs und der virtuellen Inbetriebnahme (Bild 53). In den folgenden drei Abschnitten werden diese Wege anhand der Kämmmaschine beschrieben.

4.3.1 Der Weg zum Konstruktionsentwurf

ELVAN sammelt alle noch nicht weiter spezifizierten Elementarfunktionen im Morphologischen Kasten. Aus der Funktion „Zwischenkanne nachrücken“ (Bild 58) wird somit lediglich die Funktion „Zwischenkanne führen“ zur weiteren Diskussion exportiert. Es ergibt sich der in Kapitel 3 in Bild 51 dargestellte Eintrag für diese Funktion. Die weitere Verfolgung des Materialflusses „Arbeitskanne“ führt zu der zweiten in Bild 51 aufgelisteten Funktion „Band in Arbeitskanne ablegen“. Hier wurde in ELVAN ein Sensor spezifiziert, welcher feststellt ob die Kanne mit dem gestreckten Band gefüllt ist.

In Bild 61 ist die Suche nach der geeigneten Umsetzungen dieser zwei Funktionen exemplarisch dargestellt. Die effektiv realisierte Lösung ist hier im Morphologischen Kasten nicht eingetragen, um die Rechte der Firma Rieter zu wahren.

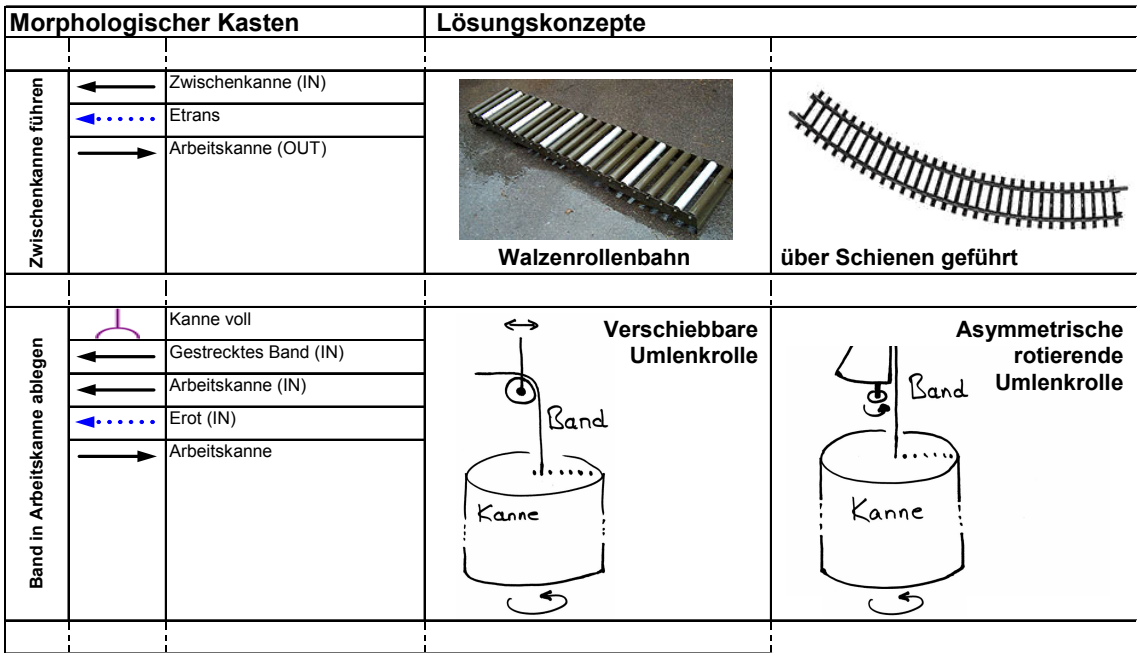


Bild 61: Lösungsvarianten für zwei Unterfunktionen der Kämmmaschine

Der Zweck der Funktion „Zwischenkanne führen“ ist es, die Kanne unter Zuhilfenahme der Translationsenergie in Richtung Arbeitsposition zu führen. Das erste Lösungskonzept sieht dazu eine passive Walzenrollenbahn vor. In der zweiten Variante wird die Kanne über Schienen geführt. Die Funktion „Band in Arbeitskanne ablegen“ wird erst relevant, wenn die Arbeitskanne korrekt positioniert ist. Das Band muss hier unter Zuhilfenahme der Rotationsenergie gleichmässig in der Kanne abgelegt werden. In beiden Lösungskonzepten ist ein lokaler Aktor zur Bewegung der Umlenkrolle vorstellbar. Fällt die Entscheidung für eine weitere Konkretisierung der zweiten Lösung mit einem lokalen Drehmotor für die asymmetrische Umlenkrolle, so führt dies zu weiteren Unter- und Aktorfunktionen mit zusätzlichen Flüssen in der EFS. Dieses Beispiel zeigt, wie der Morphologische Kasten des EFS-Tools ELVAN sukzessive zur Vervollständigung der EFS verwendet werden kann. Auch bestehende Maschinenkonzepte können so modernisiert werden, indem die Ist-EFS aufgenommen wird und mit Hilfe des Morphologischen Kastens nach neuen Lösungen gesucht wird.

Wie in Abschnitt 3.1.2 beschrieben, lässt sich die Heuristik von Stone [67] zur Modularisierung des elektromechanischen Subsystems auf die EFS anwenden. Der dominierende Fluss in der Kämmmaschine ist der Materialfluss ausgehend vom vollen Wickel bis zur vollen Kanne. Die Anwendung der Regel „Dominant flow“ (Bild 19) auf die Hauptfunktionen der Kämmmaschine (Bild 56) führt daher zu zwei potentiellen Modulen:

- Ein potentielles Modul welches die Funktion „Anlage antreiben“ beinhaltet
- Das Modul des dominanten Flusses bestehend aus allen restlichen Hauptfunktionen

Die Regel „Branching flow“ (Bild 20) führt zu einer weiteren Unterteilung des zweiten Moduls, indem der sich verzweigende Energiefluss analysiert wird. Dabei werden die Funktionen „Bänder transportieren“ und „Bänder doublieren“ zu einem potentiellen Modul zusammengefasst. Die restlichen Hauptfunktionen stellen jeweils eigene mögliche Module dar.

Die Anwendung der Regel „Conversion-transmission“ (Bild 21) führt analog zur Regel „Branching flow“ zu einer weiteren Unterteilung des dominanten Flussmodules, wobei die folgenden Funktionen den dominanten Fluss jeweils wandeln:

- Vlies bereitstellen
- Vlies kämmen
- Bänder doublieren
- Bänder strecken
- Kanne bereitstellen

Zusammen mit den Transportfunktionen ergeben sich nach der Regel „Conversion-transmission“ sechs potentielle Module wie in Bild 62 dargestellt.

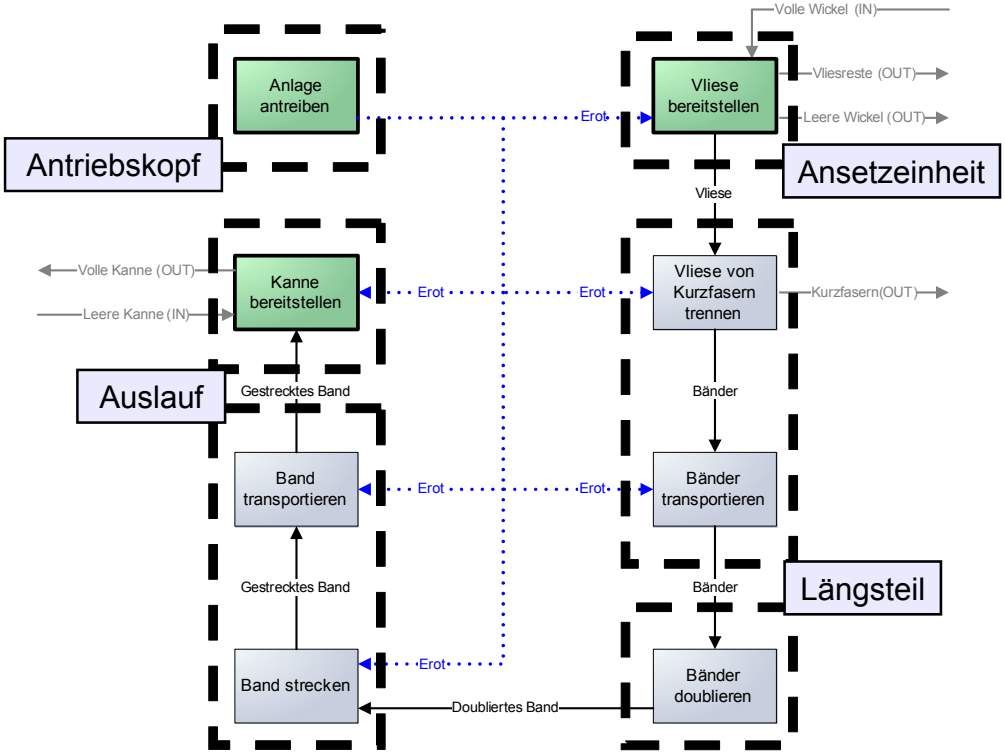


Bild 62: Module der Kämmmaschine nach der Regel „Conversion-transmission“ von Stone [67]

Alle drei Regeln der Heuristik identifizieren die Funktion „Anlage antreiben“ als eigenständiges Modul. Dies deckt sich mit der Hauptbaugruppe Antriebskopf aus Abschnitt 4.1. Die Regel „Dominant flow“ vereint die Hauptbaugruppen Ansetzeinheit, Längsteil und Auslauf zu einem potentiellen Modul. Eine feinere Unterteilung ergeben die Regeln „Branching flow“ und „Conversion-transmission“, so dass die Heuristik von Stone zwar nicht zwingend eine Modula-

risierung vorschreibt, aber gute Anhaltspunkte für eine solche gibt. In ELVAN wurden die vier Hauptbaugruppen wie in Bild 63 gezeigt angelegt.

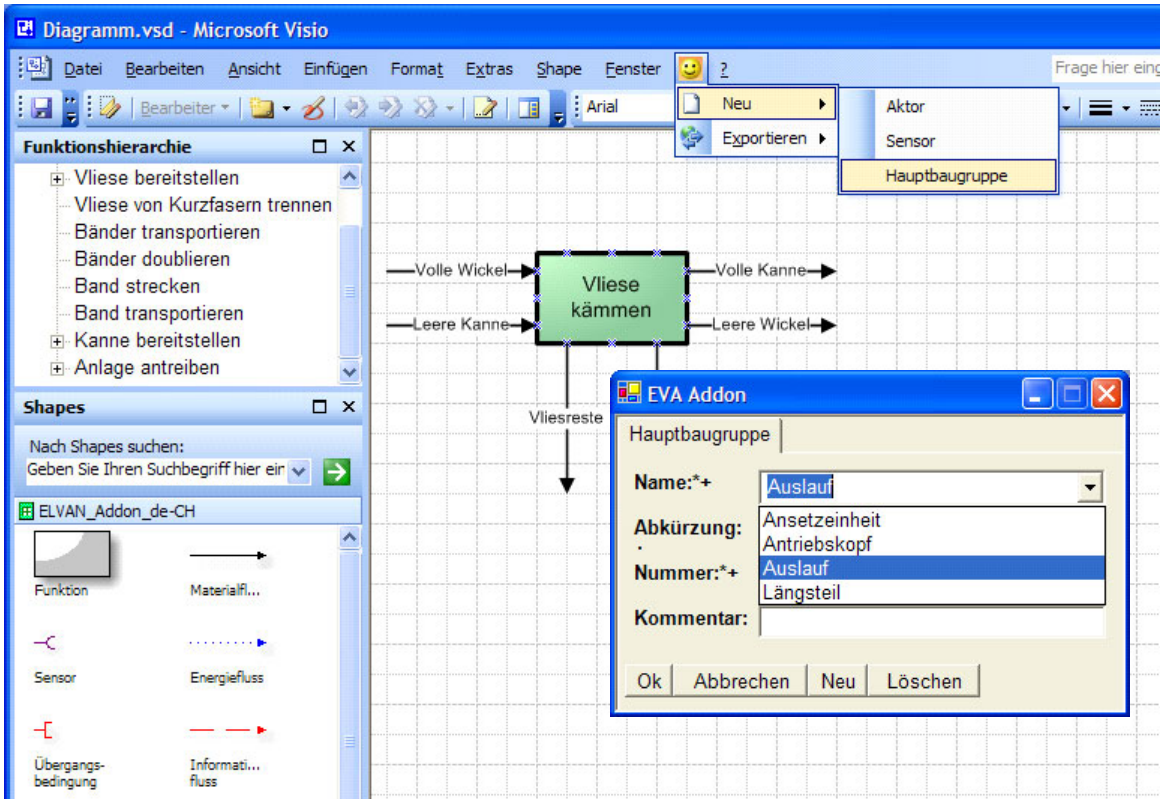


Bild 63: Hauptbaugruppen der Kämmmaschine in ELVAN

Die Partitionierung der EFS in mehrere Hauptbaugruppen erleichtert das Concurrent Engineering innerhalb der Konstruktion. Wie in Abschnitt 3.2 zu Bild 41 beschrieben, können diese Hauptbaugruppen auf mehrere Konstrukteure verteilt werden, um synchron den Konstruktionsentwurf der Maschine zu starten. Bild 64 zeigt die Hauptbaugruppe „Auslauf“, wie sie im CAD entworfen wurde.

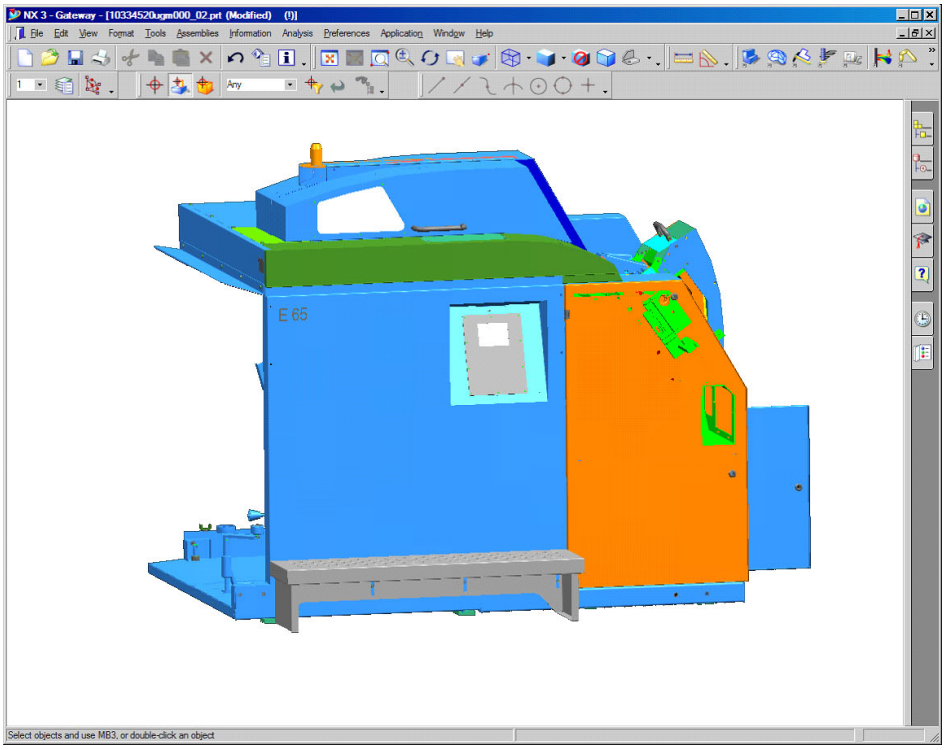


Bild 64: Der Auslauf der Kämmmaschine als CAD Modell in NX von UGS [71]

Die Hauptbaugruppen Ansetzeinheit, Längsteil und Antriebskopf wurden analog im CAD modelliert. Dieses Vorgehen erleichtert nicht nur das Concurrent Engineering innerhalb der Konstruktion. Durch das im Folgenden beschriebene automatische Ableiten des SFC und der I/O-Liste aus der EFS in ELVAN kann die Steuerungstechnik gleichzeitig mit der Konstruktion den Steuerungsentwurf starten.

4.3.2 Der Weg zum Steuerungsentwurf

ELVAN verfolgt beim Export wie in Abschnitt 3.1.2 beschrieben die Informationsflüsse und generiert jeweils eine ASCII-Datei zur Ablauflogik in SFC und zur Deklaration aller Input- und Outputvariablen in der I/O-Liste (Bild 49). Nach dem Import in die SPS-Programmierumgebung SIMATIC [65] präsentiert sich die I/O-Liste und das SFC für die Kämmmaschine wie in Bild 65 dargestellt.

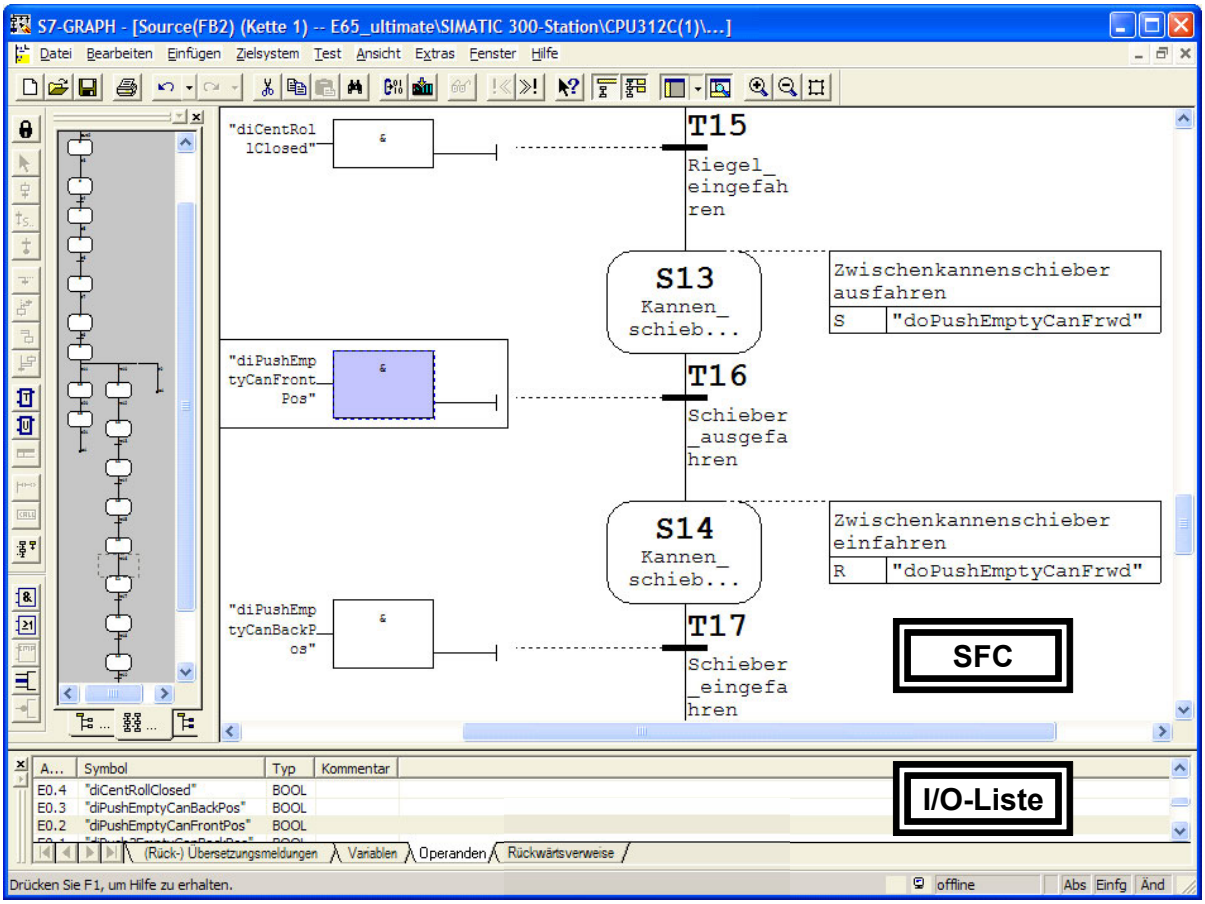


Bild 65: I/O-Liste und SFC aus ELVAN nach dem Import in SIMATIC

Im Hauptfenster ist der SFC-Ausschnitt zu der Funktion „Zwischenkanne nachrücken“ (Bild 58) visualisiert. Der Buchstabe „S“ spezifiziert in der Ablaufsprache das Setzen (Set) der Outputvariablen „doPushEmptyCanFrwd“ auf den Wert 1 (TRUE). Nachdem dieser Pneumatikzylinder ausgefahren ist (diPushEmptyCanFrontPos), wird „doPushEmptyCanFrwd“ mit der Anweisung „R“ (Reset) auf den Wert „0“ zurückgesetzt. Die Abläufe können nach dem Import in SIMATIC im Debug-Modus zusammen mit der Soft-SPS PLCSIM Schritt für Schritt durchlaufen werden (Bild 66).

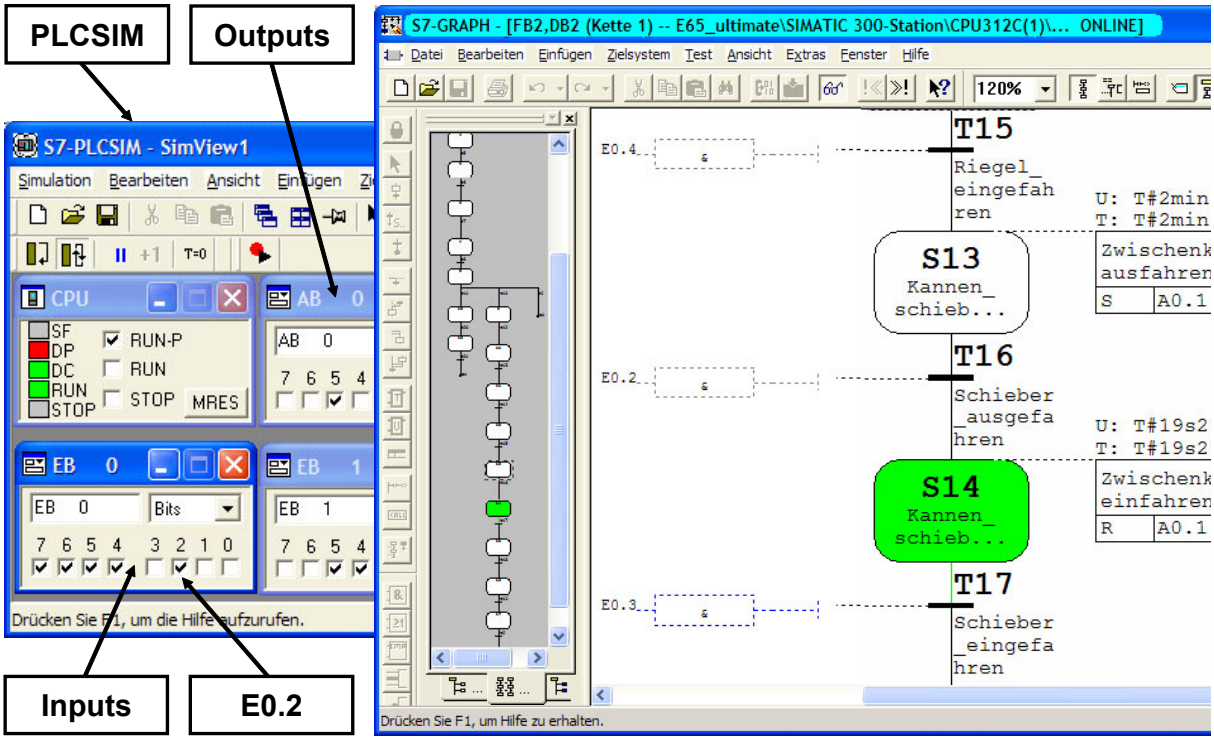


Bild 66: Die Soft-SPS PLCSIM im Zusammenspiel mit der Ablauflogik der Kämmmaschine

Der aktuelle Schritt S14 wird grün eingefärbt und das Weiterschalten zum nächsten Schritt kann durch das Ändern von Inputvariablen in der PLCSIM erzwungen werden. Dadurch werden im darauf folgenden Schritt Outputvariablen im SFC gesetzt. Das Ändern der Inputvariablen simuliert dabei den Informationseingang von den Sensoren an die SPS. Die resultierenden Änderungen für die Outputvariablen der Aktoren werden im entsprechenden Outputfenster der Soft-SPS angezeigt. Das Setzten der Inputvariablen und das Ablesen der Outputvariablen erfolgt dabei nicht über den Variablennamen, sondern über die Adressierung der Ein- und Ausgänge (Bild 6) in der SPS. So hat ELVAN z.B. für den Eingang „diPushEmptyCanFrontPos“ (Bild 59 & Bild 65) das dritte Bit im ersten Byte (E0.2) vergeben. In Bild 66 ist im Hauptfenster der Schritt S14 markiert, nachdem in der Soft-SPS der Input E0.2 gesetzt wurde.

Innerhalb der Steuerungstechnik lässt sich so das SPS-Programm in SIMATIC kompilieren und überprüfen bevor es auf die SPS geladen wird. Die Überprüfung des Steuerungsentwurfs im Zusammenspiel mit dem Konstruktionsentwurf wird anhand der Kämmmaschine im nächsten Abschnitt beschrieben.

4.3.3 Der Weg zur virtuellen Inbetriebnahme

Um das Maschinenverhalten mit einzubeziehen, müssen die Signale an die Aktoren abgefangen und in Abhängigkeit des simulierten Maschinenverhaltens Sensorsignale an die SPS gesendet werden. In diesem Fallbeispiel wurde dazu WinMOD mit PLCSIM verknüpft. ELVAN unterstützt dies durch den Export einer erweiterten I/O-Liste (Bild 52). Diese Liste enthält nicht nur die I/O‘s der Kämmmaschine, sondern auch zusätzliche Informationen zu den Aktoren und Sensoren, um das Aufsetzen der Eventsimulation zu erleichtern. In Bild 67 ist das WinMOD-Modell der Maschine zusammen mit den importierten Input- und Outputvariablen der Kämmmaschine dargestellt.

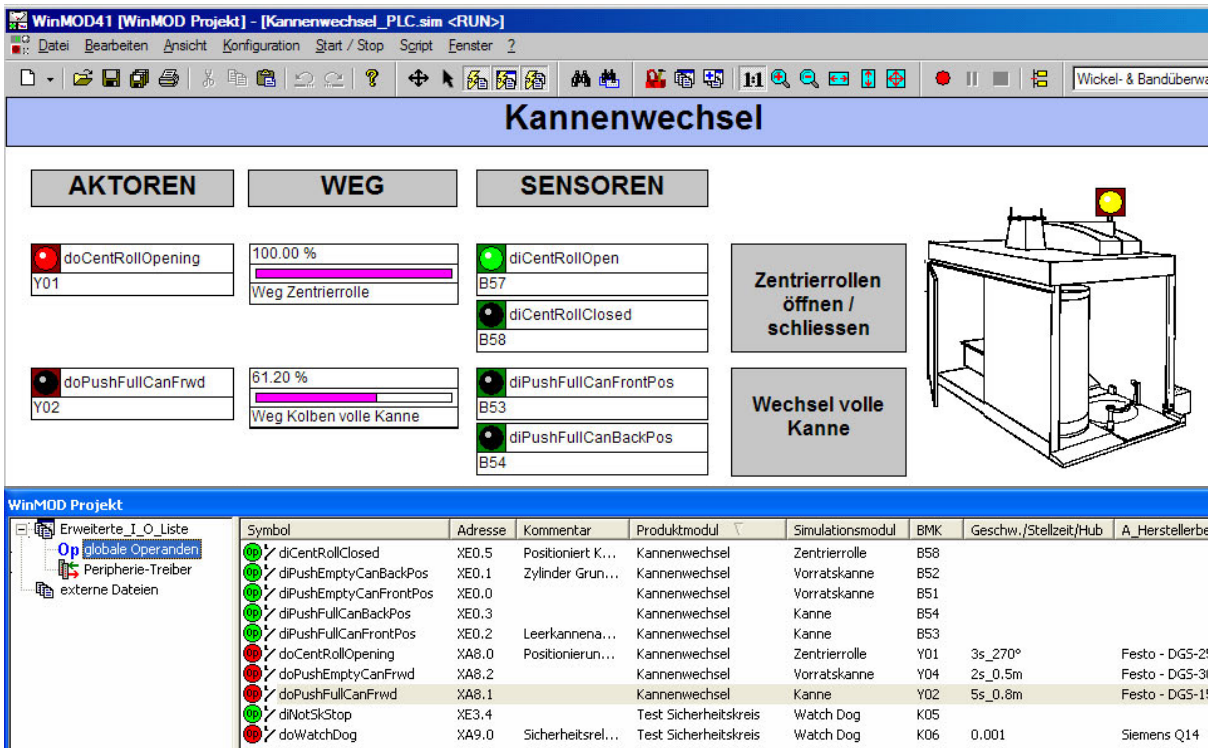


Bild 67: Abbildung des Maschinenverhaltens in WinMOD nach dem Import der erweiterten I/O-Liste

Startet man nun die Ablauflogik wie in Bild 66 dargestellt in Kombination mit WinMOD, müssen die Inputvariablen nicht mehr manuell geändert werden. Die simulierten Sensorsignale werden nun aufgrund des Maschinenmodells in WinMOD berechnet und an die Soft-SPS PLCSIM gesendet. Kombiniert man nun diese Simulation mit der Geometrie aus dem CAD

(Bild 64), erhält man eine Virtuelle Kämmmaschine nach dem in Bild 16 dargestellten Konzept. Zur Visualisierung der Maschine und ihrer Bewegungen wurde die Software Mediator [33] eingesetzt. Die Anwendung des neuen vordefinierten Prozessbausteins für die Domänenzusammenführung (Bild 43) ergibt den in Bild 68 gezeigten virtuellen Prototypen zur Überprüfung der Systemeigenschaften.

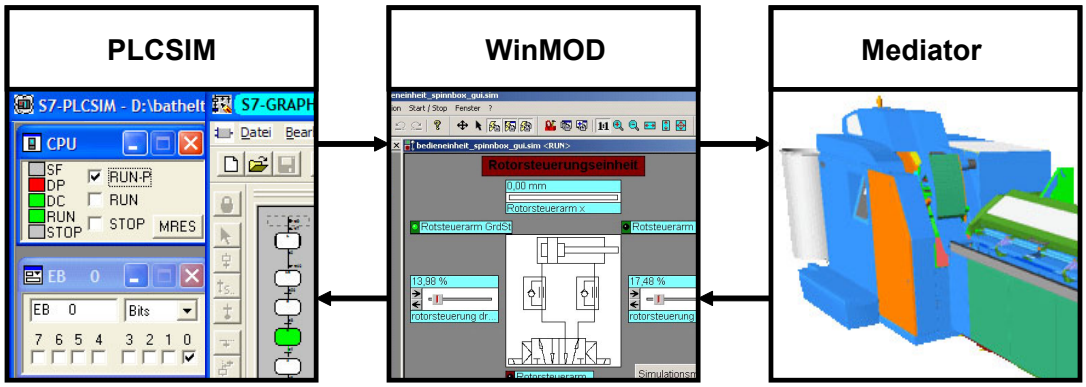


Bild 68: Die Virtuelle Kämmmaschine

Neben der Analyse der Steuerungslogik kann hier das Maschinenverhalten simuliert und visualisiert werden. Vor der realen Inbetriebnahme der Kämmmaschine testet diese Virtuelle Kämmmaschine die domänenübergreifend konsistente Umsetzung der in der EFS definierten Funktionen.

In diesem Kapitel wurde der Einsatz aller in Bild 53 gezeigten Tools beschrieben. Ausgehend von der in ELVAN definierten EFS der Kämmmaschine (Abschnitt 4.2) hat dieser Abschnitt exemplarisch aufgezeigt, wie die EFS nachfolgend zum Vorteil der Konstruktion und der Steuerungstechnik genutzt werden kann. Anregungen für weitergehende Schritte werden im nächsten Kapitel gegeben.

5 Abschliessende Betrachtungen

5.1 Diskussion

Das in Abschnitt 1.2 definierte Ziel, die Steuerungstechnik frühzeitig in den Entwicklungsprozess SPS-gesteuerter mechatronischer Systeme einzubinden, wurde mit einer EFS-basierten Entwicklungsmethodik in Kapitel 3 angegangen. Die in dieser Arbeit vorgestellte Erweiterung der Funktionsstruktur erlaubt neben der klassischen Beschreibung der konstruktiven Funktionalitäten des Systems auch die Beschreibung der SPS-Ablauflogik gemeinsam durch die Konstruktion und die Steuerungstechnik. Die Einbettung der EFS-Methoden (Abschnitt 3.1) in ein planmässiges Vorgehen (Abschnitt 3.2) strukturiert den interdisziplinären Entwicklungsprozess. In Abschnitt 3.3 wurde die neu entwickelte Software ELVAN präsentiert. Sie unterstützt das Erfassen und Nutzen der EFS. Neben dem Einstieg in die domänenspezifischen Entwurfswerkzeuge erleichtert ELVAN auch das Erstellen virtueller Prototypen.

Die konsequente Parallelisierung der Steuerungstechnik mit der Konstruktion führt zu einer Zeiteinsparung der gesamten Entwicklungsdauer im Vergleich zum traditionell sequentiellen Vorgehen, ohne per se die Aufwände zu verringern. Kosteneinsparungen lassen sich durch die Verbesserung der Konsistenz im Entwurf realisieren, indem teure Fehlerbehebungen vermieden werden (Abschnitt 1.1). Das planmässige Vorgehen aus Abschnitt 3.2 impliziert drei Mechanismen, um domänenübergreifende Inkonsistenzen gar nicht erst entstehen zu lassen oder früher zu detektieren:

- EFS in der interdisziplinären Konzeptphase
- Propagation von Änderungen im domänenspezifischen Entwurf
- Überprüfung des Komponentenentwurfs (Bild 15) durch die Virtuelle Maschine

Durch die gemeinsame Konzeption des SPS-gesteuerten mechatronischen Systems über die EFS wird eine konsistente Beschreibung erzwungen. Mit Hilfe der Software ELVAN wird die Konsistenz der Beschreibung auch bei einer Änderung der EFS sichergestellt. Weiter kann die EFS in ELVAN auch noch während des domänenspezifischen Entwurfs eingesetzt werden, um festzustellen, inwieweit eine Änderung die andere Domäne betrifft. ELVAN unterstützt ausserdem den Export einer erweiterten I/O-Liste zur Kopplung der SPS über eine Eventsimulation

mit der CAD-Geometrie. Dadurch kann schlussendlich vor der realen Inbetriebnahme die globale Konsistenz des Systems überprüft werden.

Neben der Konsistenzsicherung auch bei Änderungskonstruktionen bietet die EFS einen Ansatz, um neue Konzepte von SPS-gesteuerten mechatronischen Systemen zu entwickeln. Dabei wird dem Potential der Steuerungstechnik früher und stärker Rechnung getragen, was innovative Konzeptionen bei Neukonstruktionen erleichtert. Innovative Produkte können prinzipiell auch beim heutigen Vorgehen (Bild 2) entwickelt werden, da der Lösungsraum nicht durch das sequentielle Vorgehen eingeschränkt wird. Allerdings führt der späte Einbezug der Softwareentwicklung dazu, dass innovative interdisziplinäre Lösungen mehrere Iterationen des heutigen Vorgehens (Bild 2) bedingen. Da der Entwicklung nur ein beschränktes Zeitbudget zur Verfügung steht, hat das parallele interdisziplinäre Vorgehen (Bild 4) ein höheres Potential innovative mechatronische Produkte hervorzubringen.

Wird z.B. bei der traditionellen Konzeption in der Konstruktion eine Kurvenscheibe als Lösung einer benötigten Bewegungsfunktion fixiert und anschliessend entworfen, hätte die Steuerungstechnik schon beim Beschreiben der Funktion in der EFS eine elektronische Kurvenscheibe vorschlagen können. In modernen Steuerungen können neben den typischen SPS-Sequenzen (Abschnitt 2.3) auch komplexe Bewegungszusammenhänge über Kurvenscheiben-Editoren ([64], [7],..) programmiert werden. Eine Umsetzung der elektronischen Kurvenscheibe anstelle der schon entworfenen mechanischen Kurvenscheibe hat einen grossen Einfluss auf die entsprechende Baugruppe. Die mechanische Kurvenscheibe muss einem neuen Antriebskonzept weichen. Dies führt entweder zu einer Iteration im Entwicklungsprozess oder zur Umsetzung der mechanischen Variante.

Die EFS kann zusätzlich als domänenübergreifendes Kommunikationsmittel eingesetzt werden. Heutzutage kommen verschiedene Beschreibungsformen wie Tabellen, Ablaufdiagramme, Prozessdiagramme, hierarchische Darstellungen, Skizzen und allgemeine Flussdiagramme zum Einsatz. Dabei werden domänenspezifische Notationen verwendet. Die EFS stützt sich auf bewährte graphische Elemente, die intuitiv von der Steuerungstechnik und der Konstruktion interpretiert werden können. Dadurch erleichtert sie die Kommunikation zwischen den Domänen.

Schlussendlich dient die EFS auch der Dokumentation des SPS-gesteuerten mechatronischen Systems. Die in der EFS inhärente I/O-Liste dokumentiert die verwendete Aktorik und Sensorik. Der prinzipielle Aufbau des Systems lässt sich über die EFS jederzeit nachvollziehen. Im Gegensatz zur Umsetzung der Funktionen ändern sich die Funktionen selber selten. Je höher die Funktion in der Hierarchie steht, desto abstrakter und lösungsneutraler beschreibt sie die Funktionalität des Systems und ist somit seltener von Änderungen betroffen. Die EFS bietet somit über mehrere Maschinengenerationen eine stabile Basis zur Dokumentation des Systems.

Insgesamt ergibt sich eine Verbesserung bei der Entwicklung SPS-gesteuerter mechatronischer Systeme durch die Anwendung der Methodik aus Kapitel 3 in den folgenden Themenfeldern:

1. kürzere Entwicklungszeiten
2. kontinuierliche Konsistenz
3. innovative Konzeption
4. domänenübergreifende Kommunikation
5. stabile Dokumentation

Quantitative Aussagen zu den Themenfeldern unter Punkt 1 (kürzere Entwicklungszeiten) und Punkt 2 (kontinuierliche Konsistenz) lieferte eine Befragung von 16 in der Automatisation tätigen Unternehmen [47]. Danach wird die potentielle Verkürzung der Entwicklungsdauer von der Anforderungsliste bis zur Inbetriebnahme durch die Parallelisierung nach Kapitel 3 mit 5-10% angegeben. Dieser Punkt 1 beeinflusst in erster Linie den Faktor Zeit und nicht die Entwicklungskosten. Im Vergleich zum heutigen Vorgehen wurden die Aufwände in der Konstruktion und der Steuerungstechnik bezüglich des ersten Punktes als gleichhoch angenommen. Die potentielle Zeiteinsparung in der Entwicklung durch die verbesserte Konsistenz (zweiter Punkt) wurde mit zusätzlichen 3-10% beziffert. Dieser Punkt führt neben kürzeren Entwicklungszeiten vorallem zu Kosteneinsparungen, da sich Zeiteinsparungen wegen konsistenter Entwürfe direkt in geringeren Aufwänden wiederspiegeln. Darüber hinaus steigen die Kosten zur Bereinigung inkonsistenter Entwürfe im Verlaufe des Projekts überproportional.

Das Erstellen und Pflegen der EFS resultiert nicht zwingend zu einem Mehraufwand im vergleich zum heutigen Vorgehen. Die Steuerungstechnik konzipiert die SPS-Software schon heute mit ablauforientierten Diagrammen wie GRAFCET (Abschnitt 2.3). In der Konstruktion ist

analog dazu heute schon das Erstellen der Funktionsstruktur vorgesehen (Abschnitt 2.2). Das Erstellen der EFS ersetzt diese beiden domänenspezifischen Arbeitsschritte. Änderungen im Entwurf müssen heute manuell bewertet und kommuniziert werden, falls sie sich domänenübergreifend auswirken. Das Nachführen der Änderung in der EFS zeigt auf ob und wo diese Änderung die jeweilig andere Domäne betrifft (Abschnitt 3.1.3). Hier empfiehlt sich die konsequente Verwaltung aller EFS-Daten zusammen mit allen domänenspezifischen Entwurfsdaten auf einem PDM-System. Insgesamt werden die Vorteile des EFS-basierten Entwicklungsprozesses nicht durch zusätzliche Arbeitsschritte erkauft, sondern durch das Ersetzen schon existierender Arbeitsschritte.

5.2 Ausblick

5.2.1 Verallgemeinerung der Entwicklungsmethodik

Diese Arbeit fokussiert auf Systeme, die als Steuerung eine SPS verwenden. Gleichwohl hat der EFS-Ansatz das Potential auf andere mechatronische Systeme verallgemeinert zu werden. Die Ablauflogik der Steuerung, wie sie in der EFS enthalten ist, findet sich z.B. auch in eingebetteten Systemen (embedded systems). Die in Abschnitt 3.1.1 definierte EFS kann analog für diese Systeme eingesetzt werden. Da SFC eine typische SPS-Programmiersprache ist, muss das Ableiten eines Programms von der EFS (Abschnitt 3.1.2) für ein eingebettetes System auf die gewünschte Programmiersprache angepasst werden. Die Konstruktion kann die Methoden aus Abschnitt 3.1 unverändert übernehmen.

Die Aktorik und Sensorik stellt bei allen mechatronischen Systemen ein Bindeglied zwischen der Steuerung und des Grundsystems dar (Bild 5). Somit ist die Idee domänenübergreifende Änderungen über die Aktoren und Sensoren zu verbreiten (Abschnitt 3.1.3) auf alle mechatronischen Systeme anwendbar.

5.2.2 Rapid Virtual Prototyping

Das Verwalten der interdisziplinären Daten in ELVAN zusammen mit dem planmässigen Vorgehen (insbesondere durch den vordefinierten Prozessbaustein in Bild 43) beschleunigt das Erstellen der Virtuellen Maschine (Bild 16). Die Exportfunktion in ELVAN zur Generierung der

erweiterten I/O-Liste (Bild 52) kann direkt zur Kopplung der Eventsimulation mit der Steuerungslogik eingesetzt werden. Zusätzlich werden die notwendigen Informationen über die verwendeten Aktoren und Sensoren in der Exceltabelle bereitgestellt. Basierend auf diesen Angaben müssen die Aktor- und Sensormodelle in der Eventsimulation noch manuell ausgewählt und verknüpft werden. Hier ergibt sich eine Möglichkeit zur Automatisierung der Simulationserstellung. Bild 69 zeigt die zu simulierenden Elemente des mechatronischen Systems bei der Erstellung eines virtuellen Prototyps bestehend aus Steuerung, Simulation und Visualisierung (Bild 16).

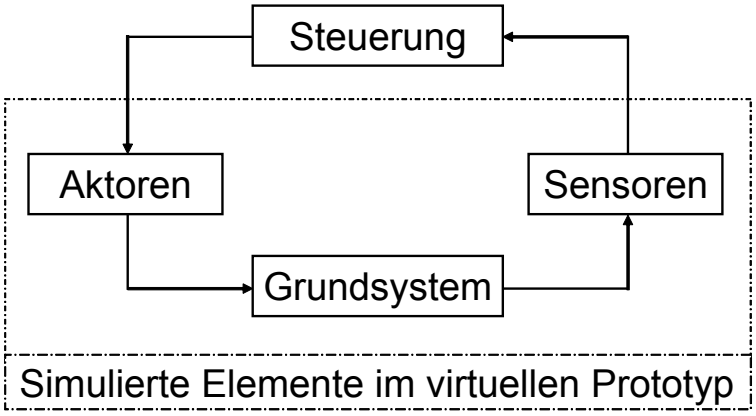


Bild 69: Simulation eines mechatronischen Systems im Rahmen der virtuellen Inbetriebnahme nach Bild 16

Die Klassifikation der Aktoren und Sensoren über die verwendete Technologie und dessen Typ (Bild 46) bietet eine Grundlage um vordefinierte Aktor- und Sensormodelle anzulegen. Wird ein konkreter Aktor ausgewählt, sind damit auch die Parameter der Aktor- und Sensormodelle definiert. Die Eventsimulationen SIMIT [66] und WinMOD [50] erlauben eine Automatisierung der Modellerstellung. SIMIT-Modelle können als XML-Dateien vordefiniert und beim Export der erweiterten I/O-Liste aus ELVAN zu einer projektspezifischen XML-Datei fusioniert werden. Ein analoges Vorgehen ist mit der aktuellen WinMOD XTE - Version 5 möglich. Mittels WinMOD-Komponentenbibliotheken und firmenspezifischen Standards können WinMOD-Projekte in wesentlichen Teilen automatisch aus den ELVAN-Daten generiert werden.

Dieses Vorgehen ermöglicht die schnelle Erstellung virtueller Prototypen zur Begutachtung der Abläufe und zur Überprüfung der korrekten Verwendung der I/O‘s. Zur Analyse des dynami-

schen Verhaltens eines mechatronischen Systems bieten sich Tools wie MATLAB/Simulink an. In [5] wird gezeigt, wie die entsprechende Virtuelle Maschine aufgebaut sein kann. Durch den modularen Aufbau der Simulation können Aktor- und Sensormodelle ausgetauscht werden, ohne dass das Grundsystem davon betroffen ist. Das automatisierte Anlegen der dynamischen Aktor- und Sensormodelle ist analog zum Vorgehen bei der Eventsimulation aus ELVAN vorstellbar.

Im Rahmen einer Dissertation wird momentan die Thematik des „rapid virtual prototyping“ an der ETH Zürich untersucht. Eine Standardisierung und Einbindung der Simulationserstellung in den Entwicklungsprozess soll den effizienten Einsatz virtueller Prototypen ermöglichen. Dabei werden neben dem CAD-Modell auch vordefinierte Simulationsmodule auf einem PDM-System verwaltet und mit den entsprechenden Baugruppen/Einzelteilen verknüpft. Basierend auf den Daten des domänenspezifischen Entwurfs kann dann bei Bedarf eine Simulation für die Virtuelle Maschine nach Dierssen [17] automatisch aus dem PDM-System abgeleitet werden.

Anhang: Benutzerhandbuch der Software ELVAN

ELVAN

Version 2.1.3

Benutzerhandbuch

Inhaltsverzeichnis

1. Einleitung 100
1.1. Was ist ELVAN? 100
1.2. Für wen ist diese Gebrauchsanweisung? 100
1.3. Konventionen innerhalb dieses Handbuchs 101
2. Installation 102
2.1. Installationsvoraussetzungen 102
2.2. ELVAN installieren 102
3. Benutzeroberfläche 104
3.1. MS Visio 104
3.2. Erweitertes Kontextmenü (rechte Maustaste) 104
3.3. Fenster Funktionshierarchie 105
3.4. Die ELVAN-Shapes 106
3.5. Dynamische Verknüpfung der ELVAN-Shapes 107
3.6. Graphische Kennzeichnung der Shape-Zustände 109
4. Erstellen der EFS in ELVAN 111
4.1. Öffnen und Speichern von EFS-Projekten 111
4.1.1. Öffnen eines neuen EFS-Projekts 111
4.1.2. Umbenennen der MS Visio-Datei Diagramm im EFS-Projekt 112
4.1.3. Öffnen eines bestehenden EFS-Projekts 112
4.1.4. Speichern eines EFS-Projekts 113
4.1.5. Speichern eines EFS-Projekts unter einem neuen Namen 113
4.2. Erstellen der Gesamtfunktion 114
4.2.1. Einfügen der Gesamtfunktion 114
4.2.2. Einfügen eines Flusses 115
4.2.3. Einfügen eines Sensors 115
4.3. Erstellen der Hauptfunktionen 116
4.3.1. Hauptfunktionen anlegen 116
4.3.2. Flüsse anlegen 117
4.3.3. Übergangsbedingungen anlegen 117
4.4. Unterfunktionen hinzufügen 117
4.4.1. Unterfunktionen anlegen 118
4.4.2. Löschen eines Arbeitsblatts von Unterfunktionen 119

4.5. Aktor definieren 119
4.5.1. Aktor-Typ und Aktor-Zustand definieren 120
4.5.2. Aktor-Outputs definieren 121
4.5.3. Aktor-Inputs definieren 122
4.5.4. Hinterlegen eines Dummy-Aktors 123
4.5.5. Aktor löschen 123
4.5.6. Aktorname im Kontextmenü 123
4.6. Sensor definieren 123
4.6.1. Sensor-Typ definieren 124
4.6.2. Sensor-Inputs definieren 125
4.6.3. Hinterlegen eines Dummy-Sensors 125
4.6.4. Sensor löschen 126
4.6.5. Sensorname im Kontextmenü 126
4.7. Logische Funktion definieren 127
4.7.1. Logische Verknüpfung bearbeiten 127
4.7.2. Inputs bearbeiten 129
4.8. SPS-ID bearbeiten 129
4.9. In der Hierarchie zeigen 130
5. ☺-Funktionen 131
5.1. ☺-Funktion Exportieren 131
5.1.1. Morphologischer Kasten 131
5.1.2. Simatic 132
5.1.3. WinMOD 134
5.1.4. Byte-Offset ändern 136

1. Einleitung

1.1. Was ist ELVAN?

ELVAN (**E**ar**L**y **V**irtual m**A**chine applicatio**N**) ist eine Software zur Erfassung und Nutzung der Erweiterten Funktionsstruktur (EFS), welche den interdisziplinären mechatronischen Produktentwicklungsprozess unterstützt. ELVAN wurde im Kontext des KTI-Forschungsprojekts EVA am Zentrum für Produktentwicklung der ETH Zürich, in Zusammenarbeit mit vier Industriepartnern – Brütsch Elektronik AG, GRITEC Institut für angewandte Technologie AG, Intelliact AG und Rieter Textile Systems AG − entwickelt. Die Besonderheit der Software besteht in der automatischen Generierung von domänenspezifischen Entwurfshilfsmitteln oder -resultaten aus der EFS:

Für die Programmierumgebung der Steuerungstechnik können die Liste der Steuerungsinputs und -outputs (I/O-Liste) und die Steuerungssoftware in Form eines Sequential Function Chart (SFC) automatisch generiert werden. Zur Unterstützung der Lösungsfindung in der Konstruktion kann ein leerer Morphologischer Kasten abgeleitet werden. Dieser enthält eine Liste der aktuellen – noch nicht weiter spezifizierten – Elementarfunktionen und deren Flüsse. Zum erleichterten, rationelleren Aufsetzen der Eventsimulation für die Virtuelle Maschine kann eine Erweiterte I/O-Liste generiert werden, welche sämtliche für die Simulationserstellung relevante Information enthält und in die Simulationsumgebung importiert werden kann. Somit steht die Software im Kontext der Hauptentwurfswerkzeuge des Computer Aided Design (CAD), der SPS-Programmierumgebung und den Tools der Virtuellen Maschine.

Als Basistool für ELVAN dient die MS Office Software *Visio*. Sie eignet sich sehr gut zur Darstellung von Funktionsstrukturen und unterstützt damit die Abbildung einer EFS. Die EFS-spezifischen Erweiterungen sind in C# als Visio Add-On realisiert. Zur konsistenten Verwaltung aller Entwurfsdaten wird *MS Access* im Hintergrund verwendet.

1.2. Für wen ist diese Gebrauchsanweisung?

Dieses Handbuch kann sowohl als Einstiegshilfe für den Erstbenutzer als auch als Nachschlagewerk für den erfahrenen Benutzer gebraucht werden. Dem Einsteiger wird empfohlen, die verschiedenen Kapitel in der vorgegebenen Reihenfolge durchzuarbeiten. Da ELVAN eine Erweiterung des Microsoft Office Programms Visio darstellt, werden die Kenntnisse von MS Visio in diesem Handbuch vorausgesetzt.

1.3. Konventionen innerhalb dieses Handbuchs

- Alle Namen und Meldungen (die von der Software angezeigt werden), sind kursiv dargestellt.
- Aufeinanderfolgende Interaktionsschritte werden durch einen Pfeil (→) getrennt.

Zum Beispiel bedeutet die Anweisung „*Datei* → *Neu* → *ELVAN*“ die Ausführung der folgenden drei Interaktionsschritte:

- Im Menü den Eintrag *Datei* wählen
- Im Untermenü den Eintrag *Neu* wählen
- Im Unter-Untermenü den Eintrag *ELVAN* wählen

Tips werden mit diesem Zeichen markiert

Hinweise werden mit diesem Zeichen markiert

Warnungen werden mit diesem Zeichen markiert

2. Installation

2.1. Installationsvoraussetzungen

- Mindestens 800 MHz Prozessor
- Mindestens 512 MB RAM
- Mindestens 4 MB freier Festplattenspeicher
- Windows XP Service Pack 2
- .Net 1.1
- Mindestens Visio Service Pack 2
- Visio 2003 mit .Net 1.1 Programmierunterstützung

2.2. ELVAN installieren

Das ELVAN Release besteht aus drei Dateien:

- EVASetup.msi
- Setup.exe
- Setup.Ini

Um die Installation zu beginnen wird die Datei *Setup.Exe* mit Doppelklick geöffnet. Darauf erscheint das Fenster des ELVAN Setup-Assistenten, welches den Benutzer über seine Schritte informiert und durch die Installation führt (Abbildung 1).

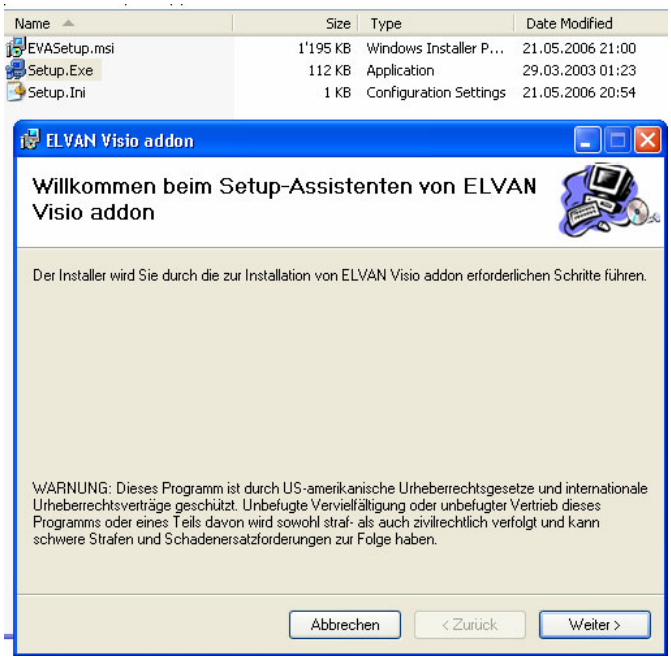


Abbildung 1: ELVAN Setup-Assistent

Die drei Schaltflächen *Abbrechen, Zurück* und *Weiter* am unteren rechten Ende der Fenster *ELVAN Visio addon* (Abbildung 1) dienen der Navigation durch die Installation, welche vier Schritte umfasst. Durch Anklicken der Schaltfläche *Abbrechen* wird die Installation

vollständig abgebrochen. Anklicken der Schaltfläche *Weiter* bringt den Benutzer in der Installation einen Schritt vorwärts und die Schaltfläche *Zurück* einen Schritt zurück.

Mit dem Beginn der Installation wird das Fenster *Installationsordner wählen* geöffnet. Darin wird der Ordner, in dem das *ELVAN Visio addon* gespeichert werden soll, festgelegt. Dazu kann durch Anklicken der Schaltfläche *Durchsuchen* ein bereits auf dem Computer bestehender Ordner ausgewählt werden. Durch Anklicken der Schaltfläche *Speicherplatzbedarf* wird der Benutzer über den zukünftig von ELVAN belegten Speicherplatz informiert. Der Benutzer kann zudem wählen, ob er das ELVAN Visio addon für alle, oder nur für den aktuellen Benutzer freigeben möchte (Abbildung 2).

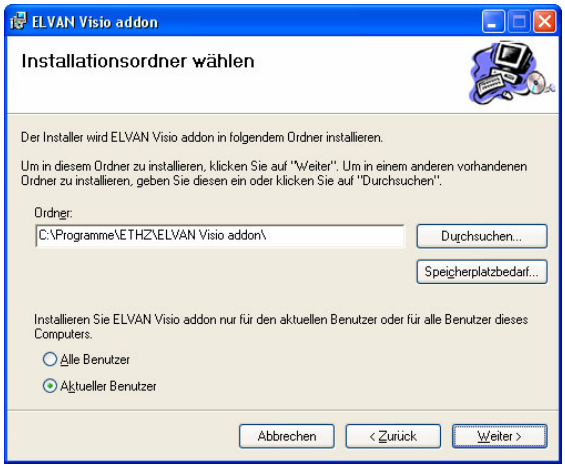


Abbildung 2: ELVAN Setup Konfiguration

Nach vollendeter Installation bestätigt der Setup-Assistent die erfolgreiche Installation von ELVAN (Abbildung 3).

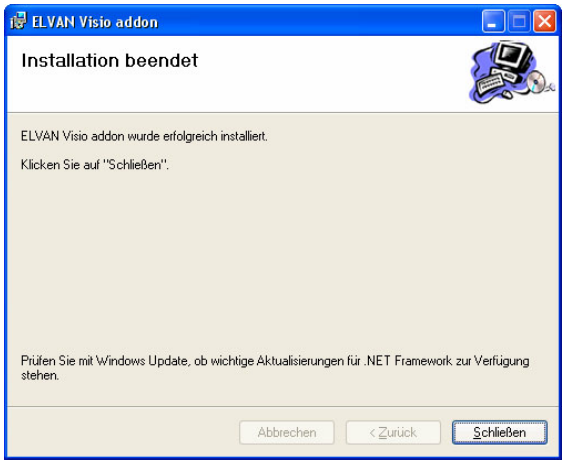


Abbildung 3: Erfolgreiche ELVAN Installation

3. Benutzeroberfläche

3.1. MS Visio

Die Menüs und Symbolleisten in MS Visio sind mit den Menüs und Symbolleisten in anderen Microsoft Office Programmen vergleichbar, so dass der Benutzer die vertraute Arbeitsumgebung vorfindet.
Die Zeichnungsumgebung von MS Visio umfasst das Zeichenblatt, Formschablonen (Shapes), Menüs und Symbolleisten. Die Zeichnung wird auf dem Zeichenblatt erstellt, welches die gedruckte Seite darstellt und ein Gitter zum Positionieren der Shapes enthält. In ELVAN finden sich zusätzlich zur Zeichnungsumgebung von MS Visio zwei Teilfenster, sowie eine Erweiterung des Kontextmenüs. Das erweiterte Kontextmenü wird durch Anklicken eines bestimmten Shapes oder des Arbeitsblatts mit der rechten Maustaste aufgerufen. Das eine Teilfenster bietet eine Übersicht über die Hierarchie in der erstellten EFS, das Andere enthält die zur Modellierung der EFS notwendigen ELVAN Shapes. Die genannten zusätzlichen Elemente von ELVAN werden in den folgenden Unterkapiteln 3.2. bis 3.4. detailliert beschrieben.

3.2. Erweitertes Kontextmenü (rechte Maustaste)

In MS Visio wird das Kontextmenü durch Mausklick (rechte Taste) auf das Arbeitsblatt geöffnet. In ELVAN wurde dieses Kontextmenü erweitert, um die Modellierung der EFS zu unterstützen. Die Erweiterung wurde im obersten Teil des Kontextmenüs angefügt, unterhalb davon ist der Fensterinhalt mit dem von MS Visio identisch. Abbildung 4 zeigt das Kontextmenü in MS Visio (links) und das erweiterte Kontextmenü in ELVAN (rechts).

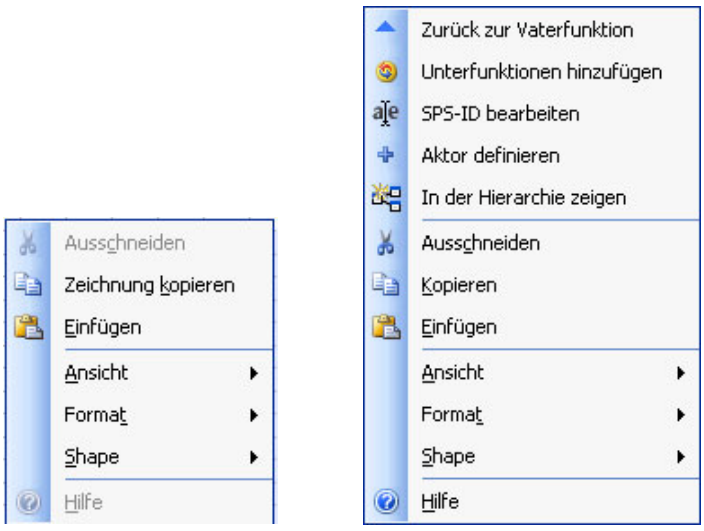


Abbildung 4: Kontextmenü in MS Visio und ELVAN

In ELVAN existieren verschiedene erweiterte Kontextmenüs. Sie wurden auf die Besonderheiten der betreffenden Shapes sowie auf die Hierarchiestufen der EFS angepasst. Wie durch den Namen bereits angedeutet, steht das Kontextmenü im Kontext der entsprechenden Hierarchiestufen, Zeichnungsblätter oder Shapes. Die Einträge sind jeweils nur vorhanden, wenn sie im Kontext der Aktivierung relevant sind.

Die auf die Hierarchiestufen und Zeichnungsblätter angepasste Erweiterung dient der vereinfachten Navigation durch dieselben. Sie lässt den Anwender durch einfaches Anklicken des Eintrags der gewünschten Hierarchiestufe auf das entsprechende Zeichnungsblatt wechseln. Dazu gehören die folgenden Einträge:

- Gehe zu den Hauptfunktionen
- Zurück zur Vaterfunktion
- Unterfunktionen anzeigen

Das Kontextmenü in Abhängigkeit der Shapes beinhaltet folgende Einträge:

- Unterfunktionen hinzufügen
- Aktor definieren
- Aktor löschen
- Aktor/*Name*
- Sensor definieren
- Sensor löschen
- Sensor/*Name*
- Logische Funktion definieren
- *Name Input*
- SPS-ID bearbeiten
- In der Hierarchie zeigen

Sämtliche obengenannten Funktionen werden in den Kapiteln 4.4. bis 4.9. beschrieben.

3.3. Fenster Funktionshierarchie

Im Fenster *Funktionshierarchie* wird die Hierarchie der EFS als Baumstruktur dargestellt. Es ist interaktiv, das heisst, durch Anklicken eines Eintrags in diesem Fenster wechselt die Ansicht automatisch auf das entsprechende Arbeitsblatt und aktiviert das entsprechende Shape. Sind dem Shape weitere Definitionen hinterlegt – wie das bei den Aktoren, Sensoren und Inputs der Fall ist – öffnet sich bei Auswahl dieser Shape-Arten zudem das shapespezifische Bearbeitungsfenster.
Die Funktionsstruktur umfasst alle Shape-Arten – ausser den drei Fluss-Typen –, sowie die den Shapes hinterlegten Aktoren, Sensoren und Inputs. Die Übergangsbedingungen werden jeweils der Funktion zugewiesen, von welcher der zugehörige Informationsfluss ausgeht. Noch nicht dynamisch verknüpfte Übergangsbedingungen und Sensoren werden in der Funktionsstruktur nicht abgebildet. Die Abbildung dieser beiden Shapes erfolgt somit erst nach erfolgter dynamischer Verknüpfung.

Abbildung 5 zeigt das Fenster Funktionshierarchie.

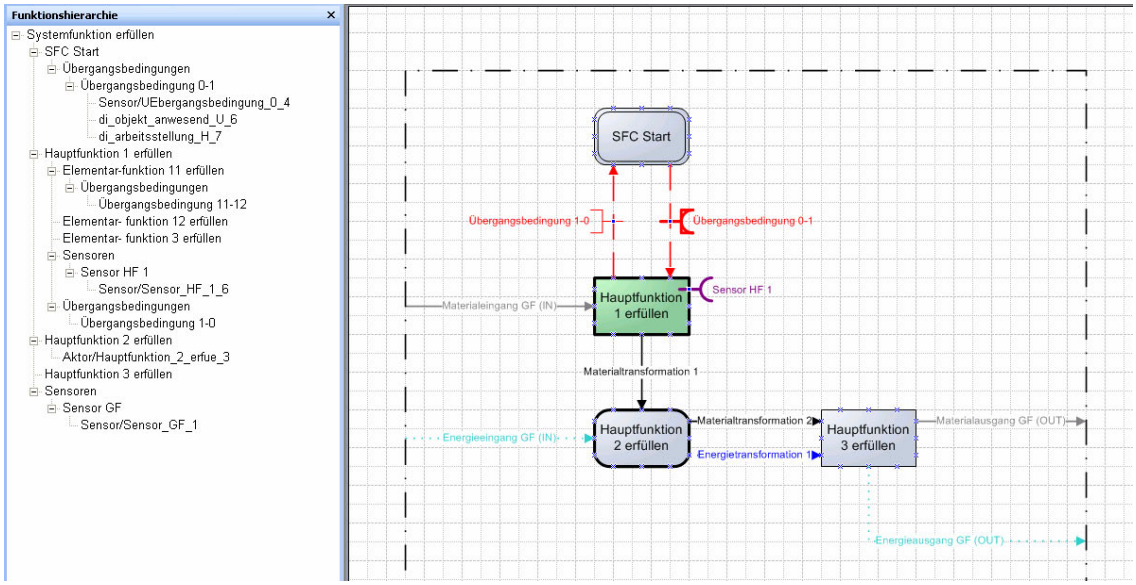


Abbildung 5: Funktionshierarchie

3.4. Die ELVAN-Shapes

In diesem Shape-Fenster finden sich alle zur Abbildung einer EFS notwendigen Formvorlagen. Es beinhaltet je ein Shape zur Abbildung der Funktionen, der Material-, Energie- und Informationsflüsse, der Übergangsbedingungen und der Sensoren. Die Shapes lassen sich einfach per drag&drop auf das Arbeitsblatt plazieren und ermöglichen damit eine einfache Erstellung der EFS. Abbildung 6 zeigt die ELVAN-Shapes.

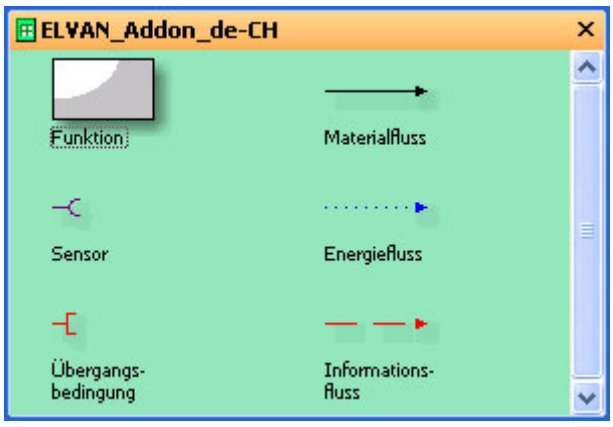


Abbildung 6: ELVAN-Shapes

Die ELVAN-Shapes lassen sich dynamisch verknüpfen. Dabei sorgt die hinterlegte Datenbank für die Datenkonsistenz in der EFS. Die dynamische Verknüpfung wird in Kapitel 3.5. beschrieben.

Zur Unterstützung einer konsistenten Modellierung der EFS werden dynamisch verknüpfte Flüsse und Übergangsbedingungen über alle Hierarchiestufen referenziert. Die referenzierten Shapes werden in ELVAN durch eine zum ursprünglichen Shape unterschiedliche Einfärbung gekennzeichnet. Neben den referenzierten Flüssen wurde in ELVAN auch eine grafische

Unterscheidung von übergeordneten Funktionen und Elementarfunktionen, hinterlegten Aktoren und Dummy-Aktoren, hinterlegten Sensoren und Dummy-Sensoren sowie die Definition von logischen Verknüpfungen oder eines Sensors bei einer Übergangsbedingung implementiert. In Kapitel 3.6. werden sämtliche Darstellungsformen der ELVAN-Shapes aufgelistet.

Durch Anklicken wird ein Shape aktiviert. Ein aktives Shape ist in Visio durch grüne, gelbe oder rote Markierungen gekennzeichnet. Die Aktivierung erlischt, durch Mausklick auf ein anderes Shape oder auf das Arbeitsblatt.

Aktivierte Shapes können mit Hilfe der folgenden vier Befehle durch Tastenkombina-tion ausgerichtet werden:

- *Ctrl+H*: Spiegeln an der vertikalen Achse
- *Ctrl+J*: Spiegeln an der horizontalen Achse
- *Ctrl+L*: 90°-Drehung im Gegenuhrzeigersinn
- *Ctrl+R*: 90°-Drehung im Uhrzeigersinn

Ein aktiviertes ELVAN-Shape kann über die Befehlstaste *F2* und anschliessender Texteingabe umbenannt werden. Gelöscht wird ein aktiviertes Shape mit der Befehlstaste *Delete*. Alternativ dazu können die Shapes auch im Fenster Funktionshierarchie nach Aktivierung des entsprechenden Eintrags auf analoge Weise umbenannt und gelöscht werden.

In ELVAN können neben den ELVAN-Shapes auch sämtliche Visio-Shapes abgebildet werden. Dies erlaubt eine Erweiterung des Modells zu Informationszwecken. Die Visio-Shapes werden in ELVAN nicht dynamisch verknüpft und verursachen somit keinen Konflikt mit der Datenbank.

Bedingt durch die geforderte Datenkonsistenz müssen die Shapes im Projekt eindeutige Namen tragen. Das gilt nicht nur für Shapes vom gleichen Typ, sondern für alle Shapes innerhalb des Projekts.

3.5. Dynamische Verknüpfung der ELVAN-Shapes

Durch die dynamische Verknüpfung wird die Datenkonsistenz in der modellierten EFS gewährleistet und die automatische Generierung der Arbeitsergebnisse aus der EFS für die Fachdisziplinen ermöglicht. Wird eine solche Verknüpfung angelegt, so löst sie einen entsprechenden Eintrag in der ELVAN hinterlegten Datenbank aus. Sie findet an den dafür vorgesehenen Stellen am Shape statt.

Folgende Verknüpfungen sind in der EFS zulässig:

- Flüsse werden ausschliesslich mit Funktionen verknüpft. Die dafür vorgesehenen Stellen an den Funktionen sind mit blauen Kreuzen (**x**) markiert. Die Flüsse werden mit ihren Anfängen (bei einem Funktionsausgang) oder mit ihren Enden (bei einem Funktionseingang) auf die dafür vorgesehen Stellen der Funktionen plaziert. Bei dynamisch verknüpften, aktivierten Flüssen ist in der Verknüpfungsstelle, ein gefülltes rotes Kästchen (■) − dem ein (+) beim Funktionseingang und ein (**x**) beim Funktionsausgang eingeschrieben ist − zu sehen. Abbildung 7 zeigt dynamisch verknüpfte

Materialflüsse, wobei der Funktionseingang aktiviert ist. In der Verknüpfungsstelle ist das rote Kästchen zu sehen.

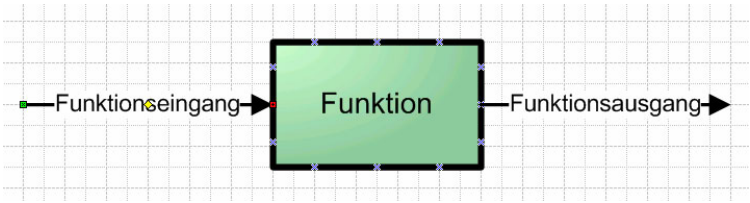


Abbildung 7: Dynamisch verknüpfte Flüsse

- Übergangsbedingungen werden ausschliesslich mit Informationsflüssen verknüpft. Die dafür vorgesehene Stelle auf dem Informationsfluss befindet sich in seiner Mitte und ist bei selektiertem Informationsfluss mit einer gelben Raute (◆), sonst mit einem blauen Kreuz (**x**) markiert. Auf den Übergangsbedingungen ist die Verknüpfungsstelle mit einem blauen Kasten (■) gekennzeichnet. Bei erfolgter dynamischer Verknüpfung erscheint das Fenster *Übergangsbedingung hinzufügen*, worin die Übergangsbedingung benannt und eine SPS-ID festgelegt werden kann. Abbildung 8 zeigt eine dynamisch verknüpfte Übergangsbedingung. Sind, wie in der Abbildung aufgezeigt, bei der Aktivierung der Übergangsbedingung die vier roten Kästchen zu sehen, so besteht die dynamische Verknüpfung.

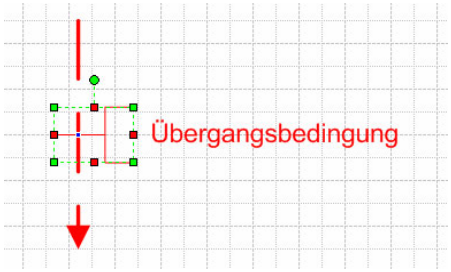


Abbildung 8: Dynamisch verknüpfte Übergangsbedingung

- Sensoren werden mit Funktionen verknüpft. Die dafür vorgesehenen Stellen auf den Funktionen sind mit blauen Kreuzen (**x**) und die auf den und Sensoren mit einem blauen Kasten (■) markiert. Bei erfolgter dynamischer Verknüpfung erscheint das Fenster *Sensor hinzufügen*, worin der Sensor benannt werden kann. Abbildung 9 zeigt einen dynamisch verknüpften Sensor.

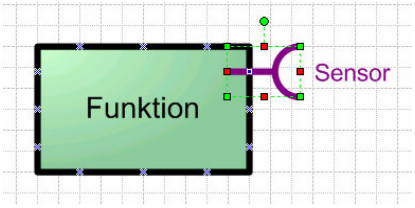


Abbildung 9: Dynamisch verknüpfter Sensor

Eine gelungene Verknüpfung wird während der Erstellung bei sämtlichen Shape-Arten mit einem nicht gefüllten roten Kästchen (□) dargestellt. Ist dieses Symbol bei der Modellierung erschienen, so wurde die dynamische Verknüpfung angelegt und in die Datenbank übertragen.

Im Laufe der Modellierung können die dynamischen Verknüpfungen, durch drag&drop der entsprechenden Shapes, wieder gelöst und geändert werden. Zur Überprüfung, ob zwischen zwei bestimmten Shapes eine dynamische Verknüpfung besteht, muss lediglich das entsprechende Shape aktiviert und auf die roten Kästchen überprüft werden.

Bei der dynamischen Verknüpfung darf die Befehlstaste *Ctrl* nicht betätigt werden. Andernfalls funktioniert ELVAN nicht mehr fehlerfrei.

3.6. Graphische Kennzeichnung der Shape-Zustände

Die graphische Kennzeichnung der Shape-Zustände dient der erleichterten Modellierung einer EFS in ELVAN. Durch sie kann der Benutzer seinen Arbeitsfortschritt sofort erkennen. Zudem wird er sich im Modell einfach zurechtfinden. Abbildung 10 bietet eine Übersicht über die in ELVAN verwendeten Darstellungsformen der Shapes und ihre Bedeutung.

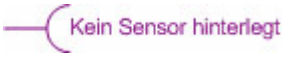
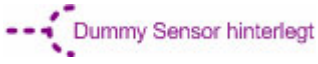
Funktionen	Funktion mit Unterfunktionen	
	Elementarfunktion ohne Aktor	
	Elementarfunktion mit Aktor (= Aktorfunktion)	
	Aktorfunktion mit Dummy-Aktor	
Flüsse	Materialfluss	
	Referenzierter Materialfluss	
	Energiefluss	
	Referenzierter Energiefluss	
	Informationsfluss	
	Referenzierter Informationsfluss	
Übergangsbedingungen	Übergangsbedingung	
	Referenzierte Übergangsbedingung	
	Übergangsbedingung mit hinterlegter logischer Verknüpfung	
	Übergangsbedingung mit hinterlegtem Sensor	
	Übergangsbedingung mit hinterlegtem Dummy-Sensor	
Sensoren	Sensor ohne Typdefinition und ohne Inputs	
	Sensor mit Typdefinition und Inputs	
	Sensor mit hinterlegtem Dummy-Sensor und Inputs	

Abbildung 10: Übersichtstabelle über die verschiedenen Shapes und ihre Abbildung

4. Erstellen der EFS in ELVAN

4.1. Öffnen und Speichern von EFS-Projekten

4.1.1. Öffnen eines neuen EFS-Projekts

1. MS Visio starten
2. In MS Visio ELVAN starten mit: Menü → *Datei* → *Neu* → *ELVAN*
3. Darauf öffnet sich das Fenster *Ordner suchen* (Abbildung 11). In diesem Fenster kann ein bestehendes Verzeichnis ausgewählt, oder mit Hilfe der Funktion *Neuen Ordner erstellen* ein neues Verzeichnis angelegt werden.
4. Mit *OK* bestätigen.

Abbildung 11: Anlegen eines Projektverzeichnisses

Wird ein bestehendes Verzeichnis angewählt, so werden die unter diesem Verzeichnis bestehenden Dateien überschrieben. Dabei erscheint ein Hinweisfenster, welches noch einmal die Bestätigung zum Überschreiben abfragt.

Damit wurde ein neues EFS-Projekt angelegt. Im ausgewählten oder neu erstellten Projektverzeichnis finden sich nun sämtliche notwendigen Projektkomponenten. Dies sind die folgenden drei Dateien:

- Diagramm in MS Visio
- Datenbank in MS Access
- Fehler Log in ASCII

In der MS Visio Datei *Diagramm* wird die EFS modelliert. Dabei sorgt die an das Diagramm verknüpfte Datenbank *EFS_Data* in MS Access für Datenkonsistenz. Im Fehler Logbuch *error_log* werden die Fehlermeldungen, die während der Modellierung auftreten aufgezeichnet. Abbildung 12 zeigt die im Projektverzeichnis angelegten Dateien.

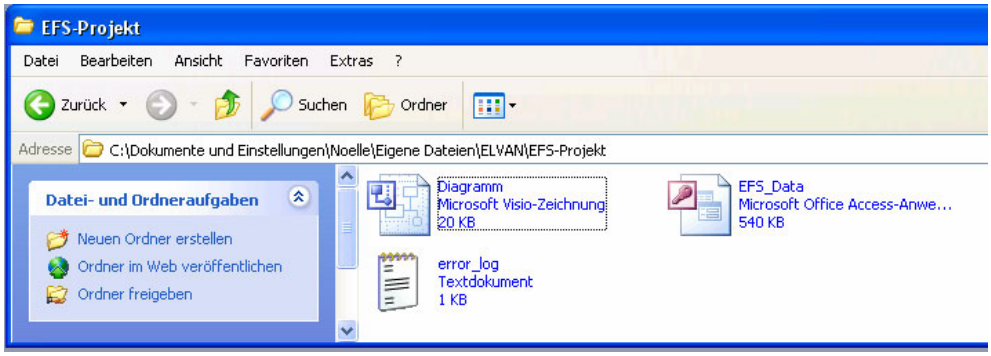


Abbildung 12: Komponenten des EFS-Projekts

Eine Umbenennung der Dateien im Projekt darf nur für die MS Visio-Datei *Diagramm* erfolgen. Dies kann über den MS Explorer vorgenommen werden.

4.1.2. Umbenennen der MS Visio-Datei Diagramm im EFS-Projekt

1. MS Explorer starten
2. Projektverzeichnis öffnen
3. Gewünschte Datei mit rechter Maustaste anklicken
4. Kontextmenü → *Umbenennen* (Abbildung 13)
   (Alternativ zu den Schritten 3. und 4. kann die Umbenennung auch über Aktivieren → Befehlstaste *F2* erfolgen)
5. Neuer Name über die Tastatur eingeben
6. Mit Enter-Taste oder linker Maustaste bestätigen

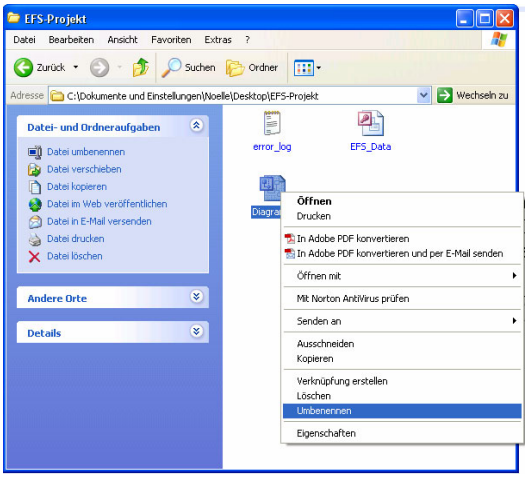


Abbildung 13: Umbenennen der Dateien des EFS-Projekts

4.1.3. Öffnen eines bestehenden EFS-Projekts

1. MS Visio starten
2. Öffnen eines bestehenden EFS-Projekts mit Menü → *Datei* → *Öffnen*

3. Darauf öffnet sich das Fenster *Öffnen* (Abbildung 14)
4. Die MS Visio Datei *Diagramm* anwählen
5. Über Schalttaste *Öffnen* oder mit Doppelklick die Datei öffnen

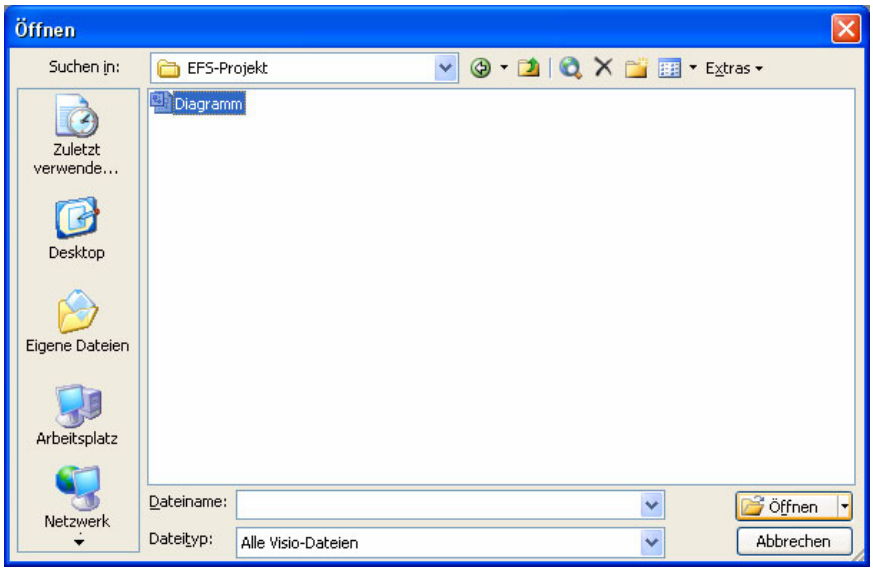


Abbildung 14: Öffnen einer bestehenden Datei

4.1.4. Speichern eines EFS-Projekts

Durch das Abspeichern des Diagramms in ELVAN mit Menü → *Datei* → *Speichern* (Abbildung 15) wird das gesamte Projekt gespeichert. Alternativ dazu kann die Tastenkombination *Ctrl+S* oder das Icon in der Menüleiste verwendet werden.

Abbildung 15: Speichern eines EFS-Projekts

Mit der Bearbeitung und dem Speichern des MS Visio Diagramms in ELVAN wird das gesamte Projekt verändert, inklusive Datenbank und Fehler Log. Für die Erstellung einer Sicherheitskopie muss aus diesem Grund das gesamte Projektverzeichnis kopiert und unter einem neuen Namen abgespeichert werden.

4.1.5. Speichern eines EFS-Projekts unter einem neuen Namen

Um ein Duplikat des Projekts unter einem neuen Namen anlegen zu können muss das gesamte Verzeichnis kopiert und neu angelegt werden (Abbildung 16). Dies wird im MS Explorer folgendermassen ausgeführt:

1. MS Explorer starten
2. Projektverzeichnis mit rechter Maustaste anklicken
3. Kontextmenü → *Kopieren*

4. Mit rechter Maustaste in das Explorer-Fenster klicken
5. Kontextmenü → *Einfügen*
6. Mit linker Maustaste bestätigen
7. Zum Umbenennen das kopierte Verzeichnis mit rechter Maustaste anklicken
8. Kontextmenü → Umbenennen
9. Neuen Namen eingeben
10. Mit Enter-Taste oder linker Maustaste bestätigen

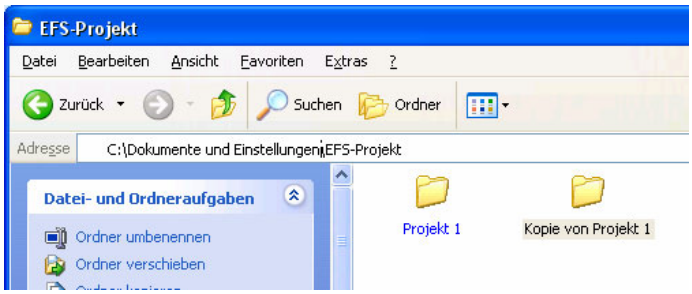


Abbildung 16: Speichern eines EFS-Projekts unter einem neuen Namen

4.2. Erstellen der Gesamtfunktion

Auf der obersten Hierarchiestufe der EFS befindet sich die Gesamtfunktion. Sie erfüllt die übergeordnete Aufgabe des abzubildenden Systems. Zusätzlich zur Gesamtfunktion können auf dieser Hierarchiestufe die Material-, Energie und Informationsflüsse, sowie eine allfällige Systemüberwachung abgebildet werden. Abbildung 17 zeigt ein Beispiel einer Gesamtfunktion mit den zugehörigen Material- und Energieflüssen, sowie einem Sensor.

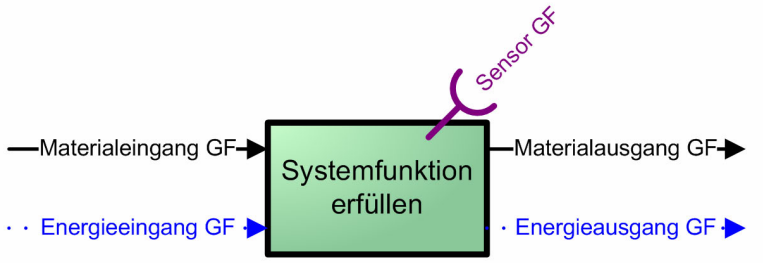


Abbildung 17: Modellierung einer Gesamtfunktion

4.2.1. Einfügen der Gesamtfunktion

Mit dem Öffnen oder neu Anlegen von ELVAN wurde das erste Arbeitsblatt *Gesamtfunktion* generiert. Darauf kann nun die Gesamtfunktion folgendermassen modelliert werden:

1. Im Fenster *ELVAN_Addon_de-CH* (Abbildung 6) das Element *Funktion* selektieren und per drag&drop auf das vorbereitete Arbeitsblatt setzen
2. Darauf öffnet sich das Fenster *Funktion hinzufügen* (Abbildung 18)
3. Gesamtfunktion benennen, durch eine entsprechende Eingabe im Feld *Text des Shapes*
4. Optional dazu kann der Gesamtfunktion auch eine Identität für die SPS-Steuerung gegeben werden. Dabei ist zu beachten, dass diese Identität nur aus maximal 24 Zeichen bestehen und keine Leerschläge oder Sonderzeichen enthalten darf
5. Mit *Ok* bestätigen

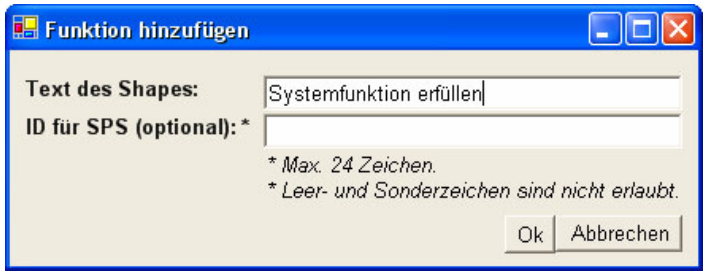


Abbildung 18: Gesamtfunktion hinzufügen

4.2.2. Einfügen eines Flusses

1. Im Fenster *ELVAN_Addon_de-CH* den gewünschten Flusstyp selektieren und per drag&drop auf das vorbereitete Arbeitsblatt setzen
2. Fluss nach Wunsch ausrichten
3. Optional kann der Fluss durch einfache Texteingabe über die Tastatur bei aktivem Shape benannt werden. Andernfalls wird ihm von ELVAN automatisch eine Nummer zugewiesen
4. Fluss mit Gesamtfunktion dynamisch verknüpfen
5. Für jeden zu modellierenden Fluss die Punkte 1. bis 4. ausführen

Die Reihenfolge der Punkte 2. bis 4. ist unwesentlich und kann nach Belieben vertauscht werden.

4.2.3. Einfügen eines Sensors

1. Im Fenster *ELVAN_Addon_de-CH* das Element *Sensor* selektieren und per drag&drop auf das vorbereitete Arbeitsblatt setzen
2. Darauf öffnet sich das Fenster *Sensor hinzufügen* (Abbildung 19)
3. Sensor benennen, durch eine entsprechende Eingabe im Feld *Text des Shapes*
4. Mit *Ok* bestätigen
5. Sensor nach Wunsch ausrichten
6. Sensor mit Gesamtfunktion dynamisch verknüpfen

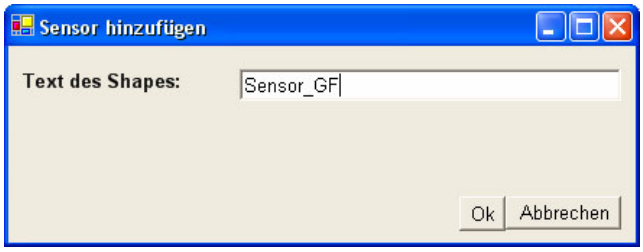


Abbildung 19: Einfügen eines Sensors

Damit wurde ein Sensor angelegt. Die Definition des Sensors sowie der zugehörigen Inputvariabeln wird in Kapitel 4.6. beschrieben.

4.3. Erstellen der Hauptfunktionen

Mit dem Erstellen der Gesamtfunktion wurde in ELVAN automatisch ein neues Arbeitsblatt angelegt. Das Arbeitsblatt, auf dem nun die Hauptfunktionen modelliert werden können. Es wird nach der Gesamtfunktion benannt, im vorliegenden Beispiel *Systemfunktion erfüllen*. Auf diesem neuen Arbeitsblatt befindet sich die Funktion *SFC Start*, welche die Initialisierung für die spätere automatische SFC-Generierung aus der EFS darstellt. Mit dieser Funktion werden ausschliesslich Informationsflüsse dynamisch verbunden. Zudem wurden auf das neu angelegte Arbeitsblatt sämtliche Flüsse, welche zur übergeordneten Gesamtfunktion gehören, übertragen und referenziert. Die referenzierten Flüsse sollen in der Folge durch den Benutzer mit den entsprechenden Hauptfunktionen dynamisch verknüpft werden. Sie sind durch ihre Färbung sowie die Zusätze (In) – bei einem Funktionseingang – oder (Out) – bei einem Funktionsausgang – zu erkennen. Damit ist eine unbeschränkte Detaillierung und somit ein sequentieller und hierarchischer Aufbau der EFS möglich. Abbildung 20 zeigt beispielhaft modellierte Hauptfunktionen mit zugehörigen lokalen und referenzierten Flüssen, Übergangsbedingungen und einem Sensor.

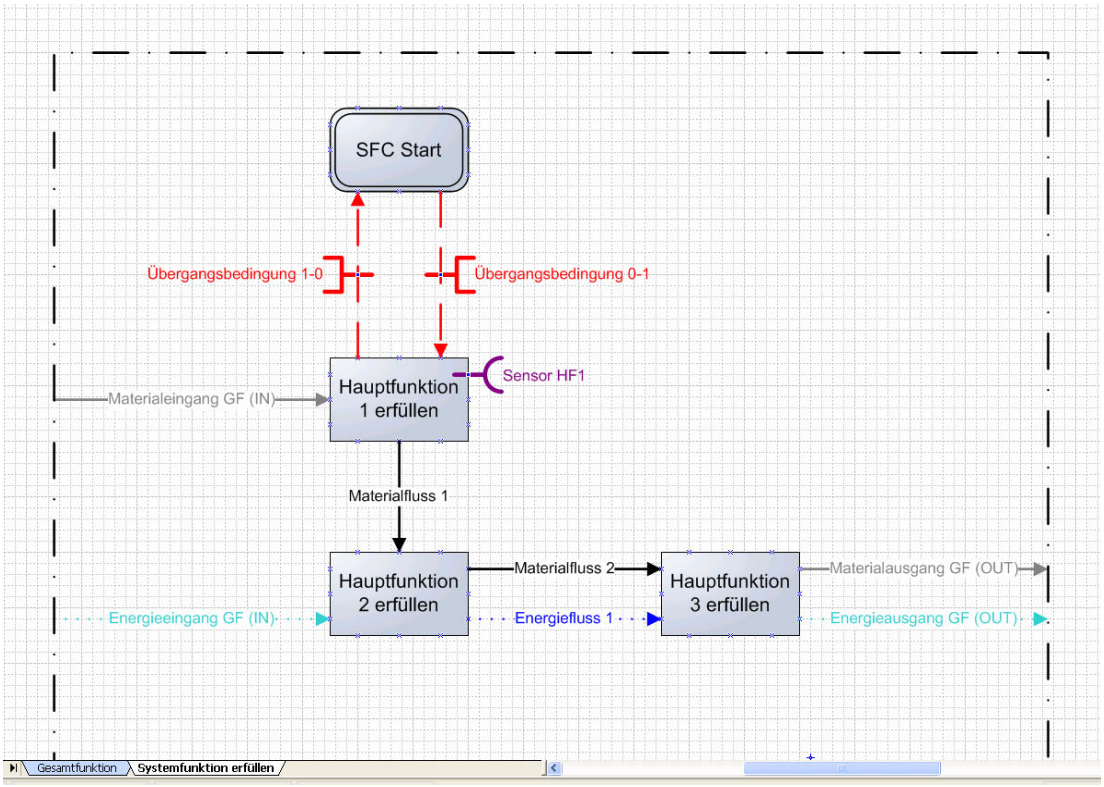


Abbildung 20: Modellierung der Hauptfunktionen

Ab dieser Hierarchieebene können nun für die Modellierung der EFS sämtliche ELVAN-Shapes 0 bis n Mal rekursiv eingesetzt werden.

4.3.1. Hauptfunktionen anlegen

Dies geschieht analog wie das Anlegen der Gesamtfunktion (Abbildung 17), nur können ab dieser zweiten Hierarchiestufe beliebig viele Funktionen modelliert werden.

1. Im Fenster *ELVAN_Addon_de-CH* das Element *Funktion* selektieren und per drag&drop auf das vorbereitete Arbeitsblatt setzen
2. Darauf öffnet sich das Fenster *Funktion hinzufügen*
3. Hauptfunktion benennen, durch eine entsprechende Eingabe im Feld *Text des Shapes*
4. Optional dazu kann der Hauptfunktion auch eine Identität für die SPS-Steuerung gegeben werden. Dabei ist zu beachten, dass diese Identität nur aus maximal 24 Zeichen bestehen und keine Leerschläge oder Sonderzeichen enthalten darf
5. Mit *Ok* bestätigen

4.3.2. Flüsse anlegen

Dies erfolgt analog der in Abschnitt 4.2.2 dargestellten Flussdefinition.

4.3.3. Übergangsbedingungen anlegen

1. Im Fenster *ELVAN_Addon_de-CH* das Element *Übergangsbedingung* selektieren und per drag&drop auf das vorbereitete Arbeitsblatt setzen
2. Darauf öffnet sich das Fenster *Übergangsbedingung hinzufügen* (Abbildung 21)
3. Übergangsbedingung benennen, durch eine entsprechende Eingabe im Feld *Text des Shapes*
4. Optional dazu kann der Übergangsbedingung auch eine Identität für die SPS-Steuerung gegeben werden. Dabei ist zu beachten, dass diese Identität nur aus maximal 24 Zeichen bestehen und keine Leerschläge oder Sonderzeichen enthalten darf
5. Mit *Ok* bestätigen
6. Übergangsbedingung nach Wunsch ausrichten
7. Übergangsbedingung mit entsprechendem Informationsfluss dynamisch verknüpfen

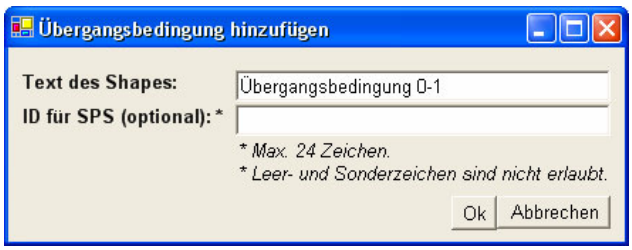


Abbildung 21: Übergangsbedingung anlegen

4.4. Unterfunktionen hinzufügen

1. Die Funktion, der Unterfunktionen hinzugefügt werden sollen, mit der rechten Maustaste anklicken.
2. Kontextmenü → *Unterfunktionen hinzufügen* (Abbildung 22)

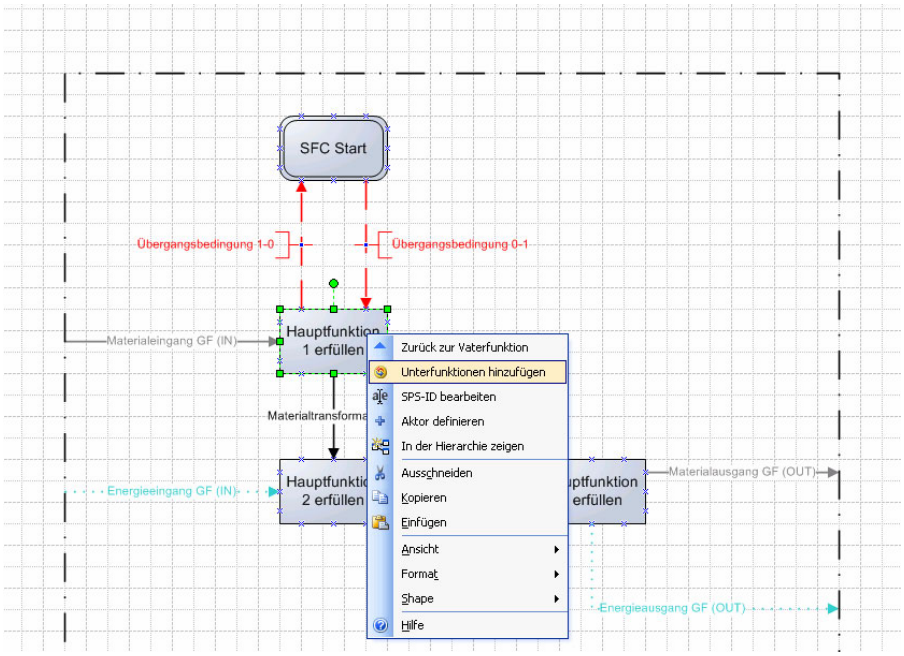


Abbildung 22: Unterfunktionen hinzufügen

Damit wurde in ELVAN automatisch ein neues Arbeitsblatt angelegt. Das Arbeitsblatt, auf dem nun die Unterfunktionen der gewählten Funktion modelliert werden können. Auch dieses Arbeitsblatt wurde nach der Vaterfunktion benannt, im vorliegenden Beispiel *Hauptfunktion 1 erfüllen*. Wiederum wurden auf das neu angelegte Arbeitsblatt sämtliche Flüsse, welche zur übergeordneten Gesamtfunktion gehören, übertragen und referenziert. Auch sollen die referenzierten Flüsse wieder mit den entsprechenden Hauptfunktionen dynamisch verknüpft werden. Bei Informationsflüssen mit dynamisch verknüpften Übergangsbedingungen, wurden diese automatisch mitreferenziert.

4.4.1. Unterfunktionen anlegen

Nun können die Unterfunktionen durch drag&drop des Elements *Funktion* definiert werden. Dies geschieht analog wie das Anlegen der Gesamtfunktion (Abbildung 17) und der Hauptfunktionen:

1. Im Fenster *ELVAN_Addon_de-CH* das Element *Funktion* selektieren und per drag&drop auf das vorbereitete Arbeitsblatt setzen
2. Darauf öffnet sich das Fenster *Funktion hinzufügen*
3. Unterfunktion benennen, durch eine entsprechende Eingabe im Feld *Text des Shapes*
4. Optional dazu kann der Unterfunktion auch eine Identität für die SPS-Steuerung gegeben werden. Dabei ist zu beachten, dass diese Identität nur aus maximal 24 Zeichen bestehen und keine Leerschläge oder Sonderzeichen enthalten darf
5. Mit *Ok* bestätigen

Den Unterfunktionen können weitere Unterfunktionen hinzugefügt werden. Somit lassen sich beliebig viele Hierarchiestufen und damit eine beliebige Detaillierung erstellen und abbilden.

4.4.2. Löschen eines Arbeitsblatts von Unterfunktionen

Das Arbeitsblatt kann nur gelöscht werden, wenn zuerst sämtliche darauf abgebildeten und definierten Unterfunktionen mit Aktivieren → *Delete* gelöscht wurden. Alternativ dazu können die Unterfunktionen auch im Fenster Funktionshierarchie auf analoge Weise gelöscht werden.

1. Durch das Hinzufügen von Unterfunktionen zu einer bestimmten Funktion wurde diese zur Vaterfunktion. Gleichzeitig wurde ein neues Arbeitsblatt erstellt, das den Namen der Vaterfunktion trägt. Soll dieses neu erstellte Arbeitsblatt gelöscht werden, so muss dies auf der Hierarchiestufe der Vaterfunktion ausgeführt werden.
2. Die Vaterfunktion mit der rechten Maustaste anklicken
3. Kontextmenü → *Aktor definieren* (Abbildung 23 links, siehe auch Kapitel 4.5.)
   (Dieser Eintrag ist im Kontextmenü nur vorhanden, wenn auf dem zu löschenden Arbeitsblatt keine Unterfunktionen vorhanden sind)
4. Darauf erscheint das Fenster *Aktor*
5. Mit *Ok* bestätigen
6. Die Vaterfunktion mit der rechten Maustaste anklicken
7. Kontextmenü → *Aktor löschen* (Abbildung 23 rechts, siehe auch Kapitel 4.5.5.)
8. Darauf erscheint das Fenster *Aktor löschen*
9. Eintrag *Aktor mit allen seinen Referenzen löschen* wählen (Abbildung 24, siehe auch Kapitel 4.5.5.)
10. Mit *Ok* bestätigen

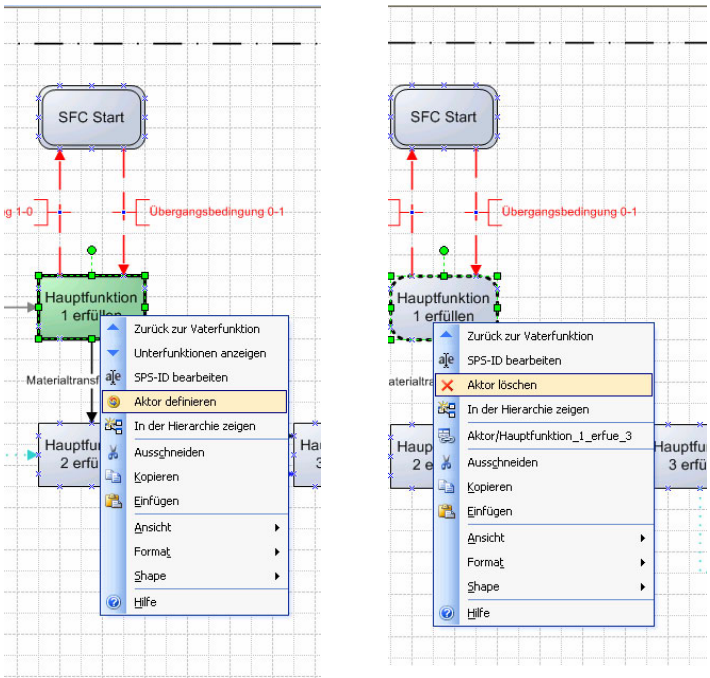


Abbildung 23: Kontextmenü: Aktor definieren/löschen

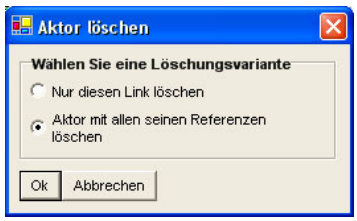


Abbildung 24: Aktor löschen

4.5. Aktor definieren

Aktoren können per Definition nur Elementarfunktionen hinzugefügt werden. Nur wenn keine Unterfunktionen bestehen, kann einer Funktion ein Aktor hinterlegt werden. Die Definition

eines Aktors im Fenster *Aktor* kann jederzeit unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden. Wird der Abbruch über die Schalttaste *Ok* ausgelöst, so werden die bis zu diesem Zeitpunkt vorgenommenen Einstellungen gespeichert. Sobald der Aktor-Typ und der Aktor-Zustand definiert wurden, setzt ELVAN automatisch eindeutige Default-Werte zur Benennung des Aktors im Projekt sowie für die Aktor-Outputs und -Inputs. Diese können bei Bedarf umbenannt werden. Um Änderungen an einem bereits definierten Aktor vorzunehmen kann das Fenster *Aktor* über den Eintrag *Aktor/Name Aktor* im Kontextmenü (Aktorfunktion mit rechter Maustaste anklicken) geöffnet werden.

Im EFS Projekt können nur Aktoren modelliert werden, welche in ELVAN bereits vordefiniert und hinterlegt wurden. Um zusätzliche Aktoren in ELVAN zu hinterlegen, müssen die in Kapitel 5 beschriebenen Schritte durchgeführt werden.

4.5.1. Aktor-Typ und Aktor-Zustand definieren

1. Die entsprechende Funktion mit der rechten Maustaste anklicken
2. Kontextmenü → *Aktor definieren* (Abbildung 23 links)
   (Dieser Eintrag ist im Kontextmenü nur vorhanden, wenn die betreffende Funktion keine Unterfunktionen aufweist und wenn der Funktion noch kein Aktor hinterlegt wurde)
3. Darauf erscheint das Fenster *Aktor*
4. Registerkarte *Typ* wählen (Abbildung 25)
5. Eintrag *Aktion* → Wählen zwischen *Neuen Aktor hinzufügen* – wenn der gewünschte Aktor im Projekt bisher keiner Funktion hinterlegt wurde – oder *bestehenden Aktor verwenden* – falls der gewünschte Aktor bereits einer anderen Elementarfunktion hinterlegt wurde.
6. Eintrag *Technologie* → Je nach Aktorausprägung die passende Technologieart anwählen
7. Eintrag *Typ* → Je nach Aktorausprägung die passende Typbeschreibung der Technologieart auswählen
8. Eintrag *Aktor* → Je nach Aktorausprägung die entsprechende Artikelnummer auswählen
9. *Zustand Aktor* → Definieren, in welchen Zustand der Aktor versetzt werden soll

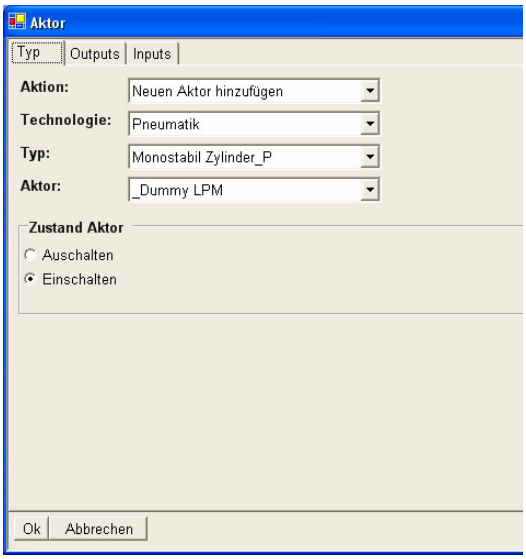


Abbildung 25: Aktor-Typ und Aktor-Zustand definieren

4.5.2. Aktor-Outputs definieren

10. Registerkarte *Outputs* wählen (Abbildung 26)
    (Gemäss der vorangegangenen Typdefinition in Kapitel 4.5.1. wurden für diese Rubrik automatisch Default-Werte gesetzt. Nach Bedarf können einige davon nun manuell verändert und ergänzt werden)
11. Übergeordneter Eintrag *Name* → In diesem Feld kann der Default-Aktorname manuell durch Aktivieren und Texteingabe geändert werden. Wichtig dabei ist, dass es sich um einen im Projekt eindeutigen Namen handelt
12. Eintrag *Wert* → Der Wert bezieht sich auf den definierten Aktor-Zustand und kann aus Konsistenzgründen ausschliesslich unter der Rubrik *Typ* verändert werden
13. Eintrag *Name* → In diesem Feld kann der Default-Outputname manuell durch Aktivieren und Texteingabe geändert werden. Wichtig dabei ist, dass es sich um einen im Projekt eindeutigen Namen handelt
14. Eintrag *Hauptbaugruppe* → Durch Aktivieren dieses Felds kann eine Hauptbaugruppe aus einer vorgegebenen Auswahl angewählt werden. (Das Hinzufügen zusätzlicher Hauptbaugruppen wird in Kapitel 5. beschrieben)
15. Eintrag *BMK* → Hier kann das Betriebsmittelkennzeichen des Outputs durch Aktivieren und Texteingabe festgelegt werden
16. Eintrag *Speicheradresse* → Beschreibt die Speicheradresse in der SPS und wird in ELVAN automatisch vergeben. Sie ist unveränderlich
17. Eintrag *Kommentar* → Hier kann ein persönlicher Kommentar des Anwenders durch Aktivieren und Texteingabe hinterlegt werden

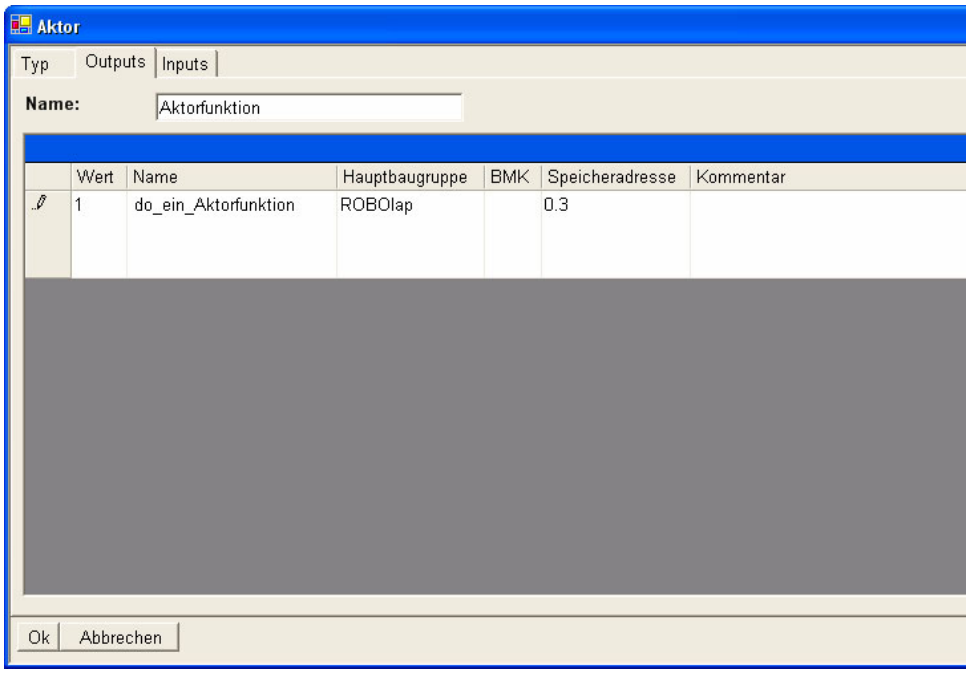


Abbildung 26: Aktor-Outputs definieren

4.5.3. Aktor-Inputs definieren

18. Registerkarte *Inputs* wählen (Abbildung 27)
    (Gemäss der vorangegangenen Typdefinition in Kapitel 4.5.1. wurden für diese Rubrik automatisch Default-Werte gesetzt. Nach Bedarf können einige davon nun manuell verändert und ergänzt werden)
19. Eintrag *Name* → In diesem Feld kann der Default-Inputname manuell durch Aktivieren und Texteingabe geändert werden. Wichtig dabei ist, dass es sich um einen im Projekt eindeutigen Namen handelt
20. Eintrag *Hauptbaugruppe* → Durch Aktivieren dieses Felds kann eine Hauptbaugruppe aus einer vorgegebenen Auswahl angewählt werden. (Das Hinzufügen zusätzlicher Hauptbaugruppen wird in Kapitel 5. beschrieben)
21. Eintrag *BMK* → Hier kann das Betriebsmittelkennzeichen des Inputs durch Aktivieren und Texteingabe festgelegt werden
22. Eintrag *Speicheradresse* → Beschreibt die Speicheradresse in der SPS und in ELVAN wird automatisch vergeben. Sie ist unveränderlich
23. Eintrag *Kommentar* → Hier kann ein persönlicher Kommentar des Anwenders durch Aktivieren und Texteingabe hinterlegt werden

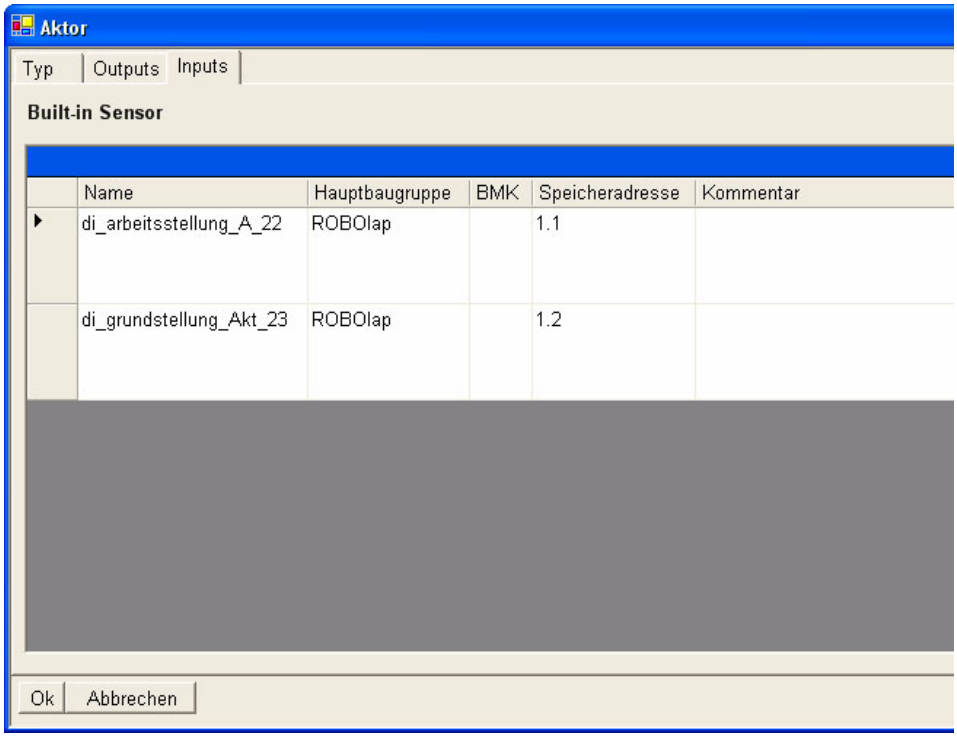


Abbildung 27: Aktor-Inputs definieren

Die optionale Detailinformation wird in die Erweiterte I/O-Liste übertragen und dient der vereinfachten Erstellung der Eventsimulation. Die Generierung der Erweiterten I/O-Liste aus ELVAN wird in Kapitel 5.2.3. beschrieben.

4.5.4. Hinterlegen eines Dummy-Aktors

Falls die Technologie und der Typ eines Aktors schon fixiert, die weiteren Attribute des Aktors aber noch unklar sind, kann ein Dummy-Aktor angelegt werden. Das Hinterlegen eines Dummy-Aktors verläuft analog zu der in den Kapiteln 4.5.1. bis 4.5.3. beschriebenen Aktordefinition. Dabei ist einzig im Fenster *Aktor* unter der Rubrik *Typ* beim Eintrag *Aktor* statt der Artikelnummer der Eintrag *Dummy* anzuwählen.

4.5.5. Aktor löschen

1. Die entsprechende Aktorfunktion mit der rechten Maustaste anklicken
2. Kontextmenü → *Aktor löschen* (Abbildung 23 rechts)
3. Darauf erscheint das Fenster *Aktor löschen* (Abbildung 24)
4. Löschungsvariante wählen: *Nur diesen Link löschen* oder *Aktor mit allen seinen Referenzen löschen*. (Dabei löscht der erste Befehl nur die Verknüpfung zwischen dem gewählten Aktor und der zugehörigen Funktion. Mit dem zweiten Befehl wird der gewählte, im Projekt definierte Aktor ganz aus selbigem gelöscht)
5. Mit *Ok* bestätigen

4.5.6. Aktorname im Kontextmenü

Durch Anklicken der Aktorfunktion mit der rechten Maustaste erscheint das Kontextmenü mit dem Eintrag *Aktor/Name Aktor*. Anklicken dieses Eintrags mit der linken Maustaste öffnet das Fenster Aktor, worin die bestehenden Aktordefinitionen geändert und erweitert werden können. (Abbildung 28) zeigt den Aktornamen im Kontextmenü:

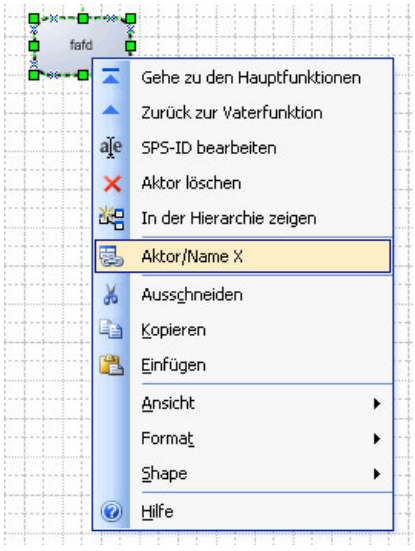


Abbildung 28: Aktornamen im Kontextmenü

4.6. Sensor definieren

Sensoren können sowohl den Übergangsbedingungen als auch den Sensor-Shapes hinzugefügt werden. Die Definition eines Sensors im Fenster *Sensor* kann jederzeit unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden. Wird der Abbruch über die Schalttaste *Ok* ausgelöst, so werden die bis zu diesem Zeitpunkt vorgenommenen Einstellungen abgespeichert. Sobald der Sensor-Typ definiert wurde, setzt ELVAN automatisch eindeutige Default-Werte zur Benennung des Sensors im Projekt sowie für die

Sensor-Inputs. Diese können bei Bedarf umbenannt werden. Um Änderungen an einem bereits definierten Sensor vorzunehmen, kann das Fenster *Sensor* über den Eintrag *Sensor/Name Sensor* im Kontextmenü (Sensor oder Übergangsbedingung mit rechter Maustaste anklicken) geöffnet werden.

Im EFS Projekt können nur Sensoren modelliert werden, welche in ELVAN bereits vordefiniert und hinterlegt wurden. Um zusätzliche Sensoren in ELVAN zu hinterlegen, müssen die in Kapitel 5. beschriebenen Schritte durchgeführt werden.

Wird der Sensor einer Übergangsbedingung hinzugefügt, so öffnet sich automatisch das Fenster des Logikeditors. Darin kann die logische Verknüpfung sämtlicher zur Übergangsbedingung gehöriger Inputs erstellt werden. Der Logikeditor wird in Kapitel 4.7. beschrieben.

4.6.1. Sensor-Typ definieren

1. Das entsprechende Sensor-Shape oder die entsprechende Übergangsbedingung mit der rechten Maustaste anklicken
2. Kontextmenü → *Sensor definieren*
   (Dieser Eintrag ist im Kontextmenü nur vorhanden, wenn dem entsprechenden Shape noch kein Sensor hinterlegt wurde)
3. Darauf erscheint das Fenster *Sensor*
4. Registerkarte *Typ* wählen (Abbildung 29)
5. Eintrag *Aktion* → Wählen zwischen *Neuen Sensor hinzufügen* – wenn der gewünschte Sensor im Projekt bisher keiner Funktion hinterlegt wurde – oder *bestehenden Sensor verwenden* – falls der gewünschte Sensor im Projekt bereits einem anderen Shape hinterlegt wurde.
6. Eintrag *Technologie* → Je nach Sensorausprägung die passende Technologieart anwählen
7. Eintrag *Typ* → Je nach Sensorausprägung die passende Typbeschreibung der Technologieart auswählen
8. Eintrag *Sensor* → Je nach Sensorausprägung die entsprechende Artikelnummer auswählen

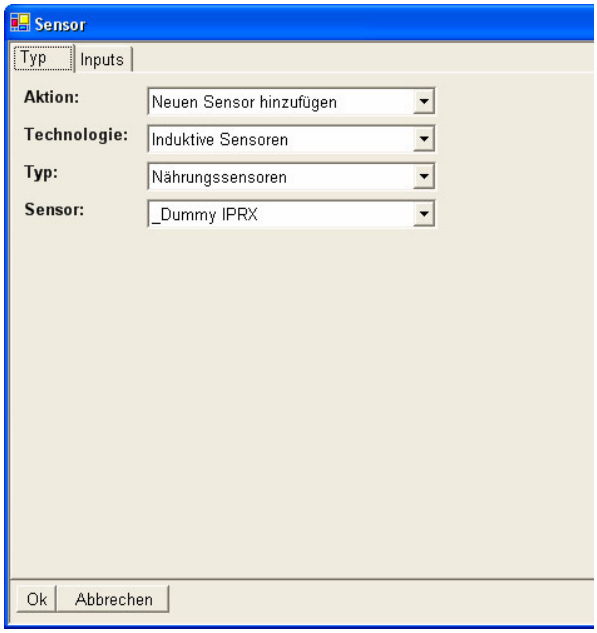


Abbildung 29: Sensor-Typ definieren

4.6.2. Sensor-Inputs definieren

9. Registerkarte *Inputs* wählen (Abbildung 30)
   (Gemäss der vorangegangen Typdefinition in Kapitel 4.6.1. wurden für diese Rubrik automatisch Default-Werte gesetzt. Nach Bedarf können nun einige davon manuell verändert und ergänzt werden)
10. Übergeordneter Eintrag *Name* → In diesem Feld kann der Default-Sensorname manuell durch Aktivieren und Texteingabe geändert werden. Wichtig dabei ist, dass es sich um einen im Projekt eindeutigen Namen handelt
11. Eintrag *Name* → In diesem Feld kann der Default-Inputname manuell durch Aktivieren und Texteingabe geändert werden. Wichtig dabei ist, dass es sich um einen im Projekt eindeutigen Namen handelt
12. Eintrag *Hauptbaugruppe* → Durch Aktivieren dieses Felds kann eine Hauptbaugruppe aus einer vorgegebenen Auswahl angewählt werden. (Das Hinzufügen zusätzlicher Hauptbaugruppen wird in Kapitel 5. beschrieben)
13. Eintrag *BMK* → Hier kann das Betriebsmittelkennzeichen des Inputs durch Aktivieren und Texteingabe festgelegt werden
14. Eintrag *Speicheradresse* → Beschreibt die Speicheradresse in der SPS und wird in ELVAN automatisch vergeben. Sie ist unveränderlich
15. Eintrag *Kommentar* → Hier kann ein persönlicher Kommentar des Anwenders durch Aktivieren und Texteingabe hinterlegt werden

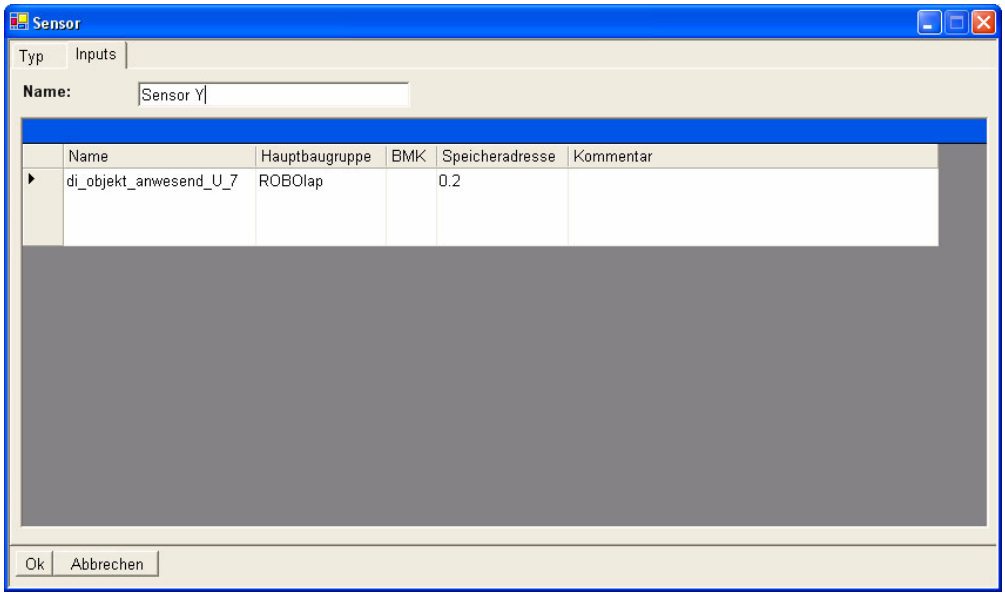


Abbildung 30: Sensor-Inputs definieren

4.6.3. Hinterlegen eines Dummy-Sensors

Falls die Technologie und der Typ eines Sensors schon fixiert, die weiteren Attribute des Sensors aber noch unklar sind, kann ein Dummy-Sensor angelegt werden. Das Hinterlegen eines Dummy-Sensors verläuft analog zu der in den Kapiteln 4.6.1. bis 4.6.2 beschriebenen Sensordefinition. Dabei ist einzig im Fenster *Sensor* unter der Rubrik *Typ* beim Eintrag *Sensor* statt der Artikelnummer der Eintrag *Dummy* anzuwählen.

4.6.4. Sensor löschen

1. Das entsprechende Shape mit der rechten Maustaste anklicken
2. Kontextmenü → *Sensor löschen* (Abbildung 31 links)
3. Darauf erscheint das Fenster *Sensor löschen* (Abbildung 31 rechts)
4. Löschungsvariante wählen: *Sensor mit allen seinen Referenzen löschen* oder *Nur diesen Link löschen* (Dabei löscht der erste Befehl nur die Verknüpfung zwischen dem gewählten Sensor und der zugehörigen Funktion. Mit dem zweiten Befehl wird der gewählte, im Projekt definierte Sensor ganz aus selbigem gelöscht)
5. Mit *Ok* bestätigen

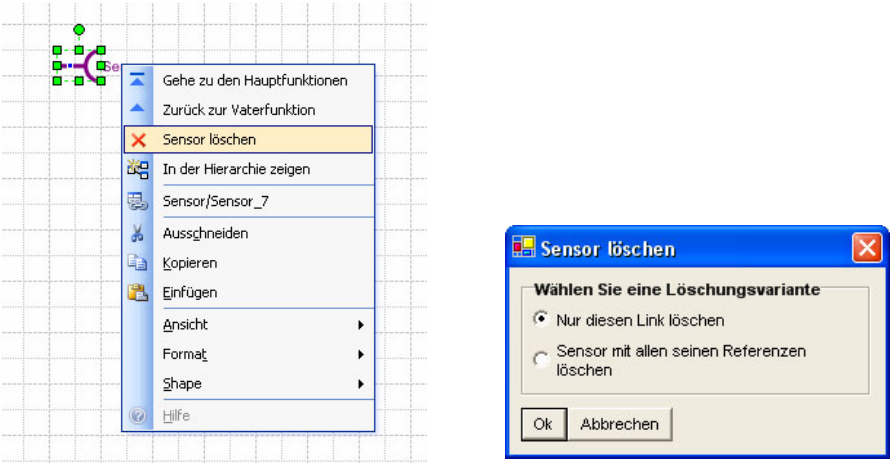


Abbildung 31: Sensor löschen

4.6.5. Sensorname im Kontextmenü

Durch Anklicken des entsprechenden Shapes mit der rechten Maustaste erscheint das Kontextmenü mit dem Eintrag *Sensor/Name Sensor*. Anklicken dieses Eintrags mit der linken Maustaste öffnet das Fenster Sensor, worin die bestehenden Sensordefinitionen geändert und erweitert werden können. (Abbildung 32) zeigt den Sensornamen im Kontextmenü.

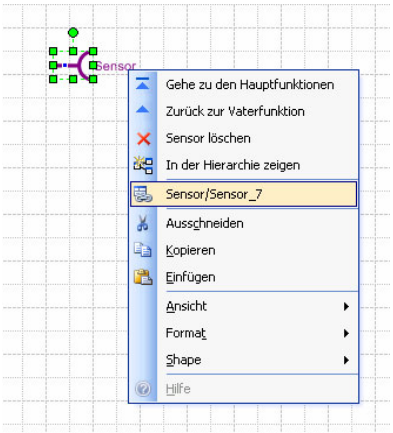


Abbildung 32: Sensorname im Kontextmenü

4.7. Logische Funktion definieren

Für die automatische Generierung des SFC in ELVAN müssen die Inputs logisch verknüpft werden. Dies geschieht im Logikeditor. Wird einer Übergangsbedingung ein Sensor hinzugefügt, so öffnet sich das Fenster des Logikeditors automatisch. Andernfalls kann der Logikeditor über das Kontextmenü geöffnet werden. Die Bearbeitung der Inputs kann jederzeit unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden. Mit der Schalttaste *Ok* werden sämtliche bisherigen Einträge gespeichert. Ein erneutes Öffnen des Logikeditors kann jederzeit über das Kontextmenü erfolgen:

1. Die entsprechende Übergangsbedingung mit der rechten Maustaste anklicken
2. Kontextmenü → *Logische Funktion definieren* (Abbildung 33 links)
3. Darauf erscheint das Fenster *Wählen Sie zwei Inputs* (Abbildung 33 rechts)
4. Eintrag *Gatter* → Art der logischen Verknüpfung wählen (*UND* oder *ODER*)
5. Einträge *Input 1* und *Input 2* →Gewünschte Inputs auswählen.
   (Mit der Auswahl des Eintrags *Kein* wird nur ein Input logisch verknüpft.)
6. Eintrag *Negieren* → Anklicken um den zugehörigen Input zu negieren
7. Mit *Ok* bestätigen
8. Darauf erscheint das Fenster *Logikeditor*
   (Hier kann die bereits getroffene Auswahl weiter präzisiert werden, Kapitel 4.7.1. und 4.7.2.)

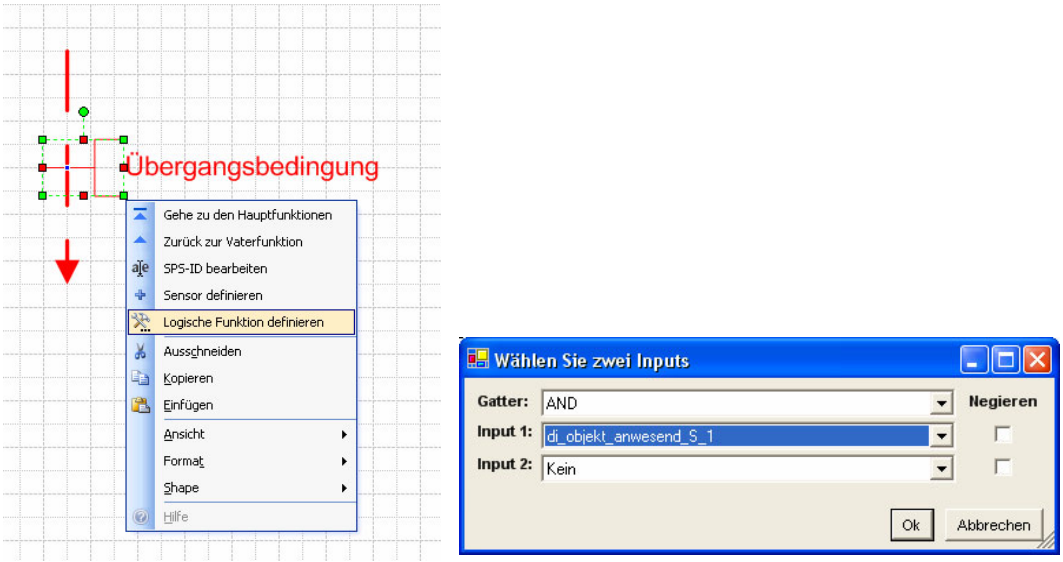


Abbildung 33: Logische Funktion definieren

4.7.1. Logische Verknüpfung bearbeiten

9. Durch Mausklick die Logikfunktion aktivieren
10. Mit rechtem Mausklick das Kontextmenü des Logikeditors auswählen
    (Abbildung 34)
11. Eintrag *Ändern* → Durch Anklicken dieses Eintrags ändert die Art der logischen Verknüpfung von *UND* auf *ODER* und umgekehrt
12. Eintrag *Löschen* → Durch Anklicken dieses Eintrags wird die aktivierte Funktion gelöscht
13. Eintrag *Ast hinzufügen* → Durch Anklicken dieses Eintrags wird der Funktion ein Ast für einen zusätzlichen Input hinzugefügt

14. Eintrag *AND-Gatter hinzufügen* → Durch Anklicken dieses Eintrags wird der ursprünglichen Logikfunktion eine weitere *UND*-Logikfunktion hinzugefügt
15. Eintrag *OR-Gatter hinzufügen* → Durch Anklicken dieses Eintrags wird der ursprünglichen Logikfunktion eine weitere *ODER*-Logikfunktion hinzugefügt

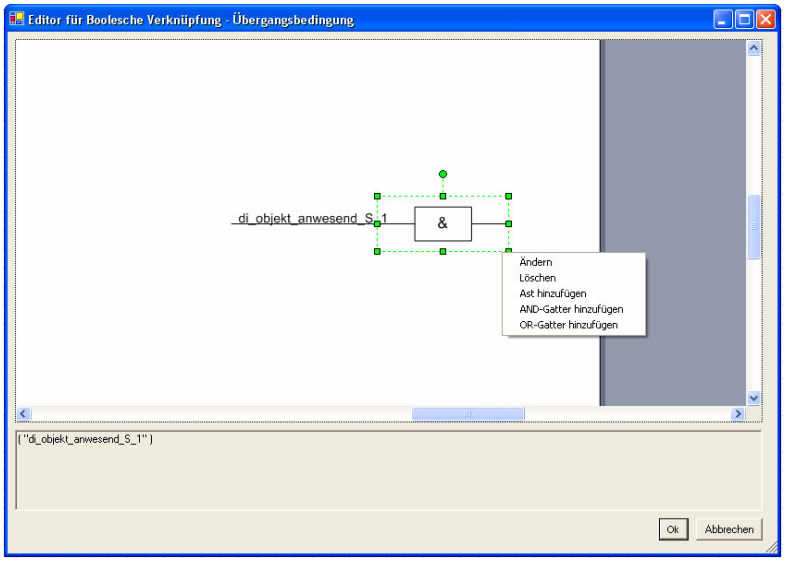


Abbildung 34: Kontextmenü der Logikfunktion

Im Logikeditor können beliebig viele Logikfunktionen miteinander verknüpft werden. Abbildung 35 zeigt ein Beispiel für eine komplizierte logische Verknüpfung.

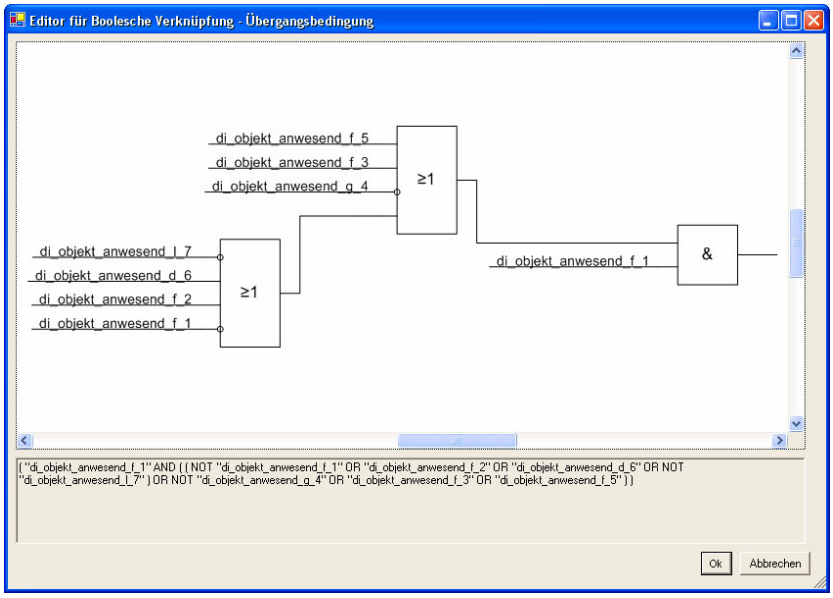


Abbildung 35: Komplizierte logische Verknüpfung

4.7.2. Inputs bearbeiten

16. Bei aktivierter Funktion durch Mausklick auf den Input-Ast diesen aktivieren
17. Mit rechtem Mausklick auf den Input-Ast das Kontextmenü der Inputs auswählen (Abbildung 36)
18. Eintrag *Ändern* → Durch Anklicken dieses Eintrags öffnet sich das Fenster *Inputs*. Darin kann der gewünschte Input ausgewählt werden
19. Eintrag *Löschen* → Durch Anklicken dieses Eintrags wird der Input mitsamt seinem Ast gelöscht
20. Eintrag *Negieren* → Durch Anklicken dieses Eintrags wird der Input negiert.
21. Mit *Ok* bestätigen

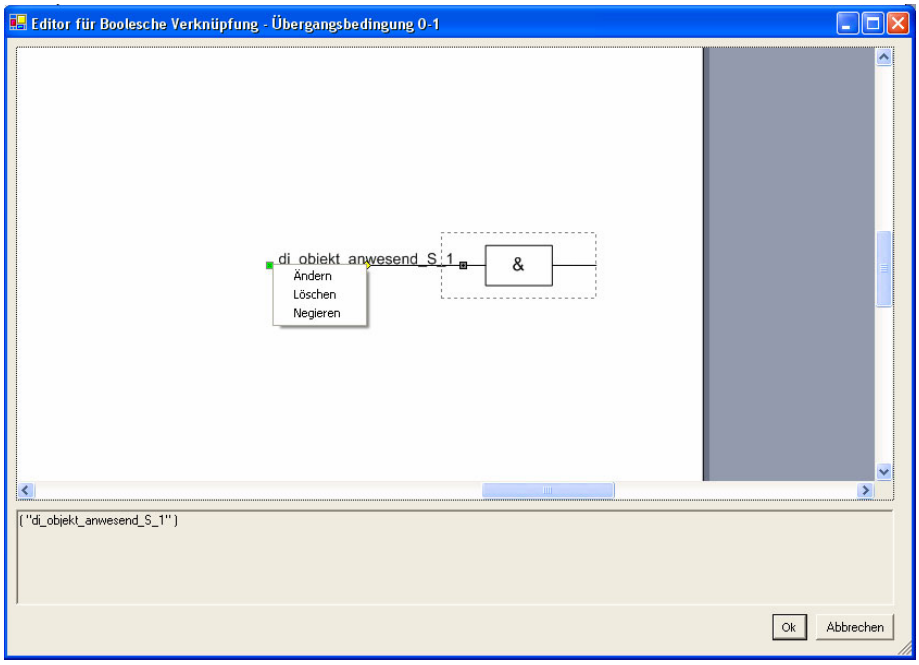


Abbildung 36: Kontextmenü der Inputs

4.8. SPS-ID bearbeiten

Mit Hilfe dieser Funktion im Kontextmenü kann die Identität der Funktionen und Übergangsbedingungen in der SPS geändert werden. Mit dem Erstellen der Funktionen und Übergangsbedingungen bestand im Fenster *Funktion hinzufügen* beziehungsweise *Übergangsbedingung hinzufügen* die Möglichkeit, die SPS Identität des Shapes zu definieren. Wurde in dieser Zeile kein Eintrag erstellt, so wurde von ELVAN automatisch eine Default-SPS-ID für das Shape definiert. Dieser Eintrag kann jedoch jederzeit über das Kontextmenü – durch Anklicken der entsprechenden Funktion oder Übergangsbedingung mit der rechten Maustaste – geändert werden.

4.9. In der Hierarchie zeigen

Durch Aktivieren dieser Funktion im Kontextmenü wird im Fenster *Funktionshierarchie* das betreffende Shape angezeigt (Abbildung 37).

1. Entsprechendes Shape mit der rechten Maustaste anklicken
2. Kontextmenü → *In der Hierarchie zeigen*
3. Im Fenster *Funktionshierarchie* wird der entsprechende Eintrag blau hervorgehoben

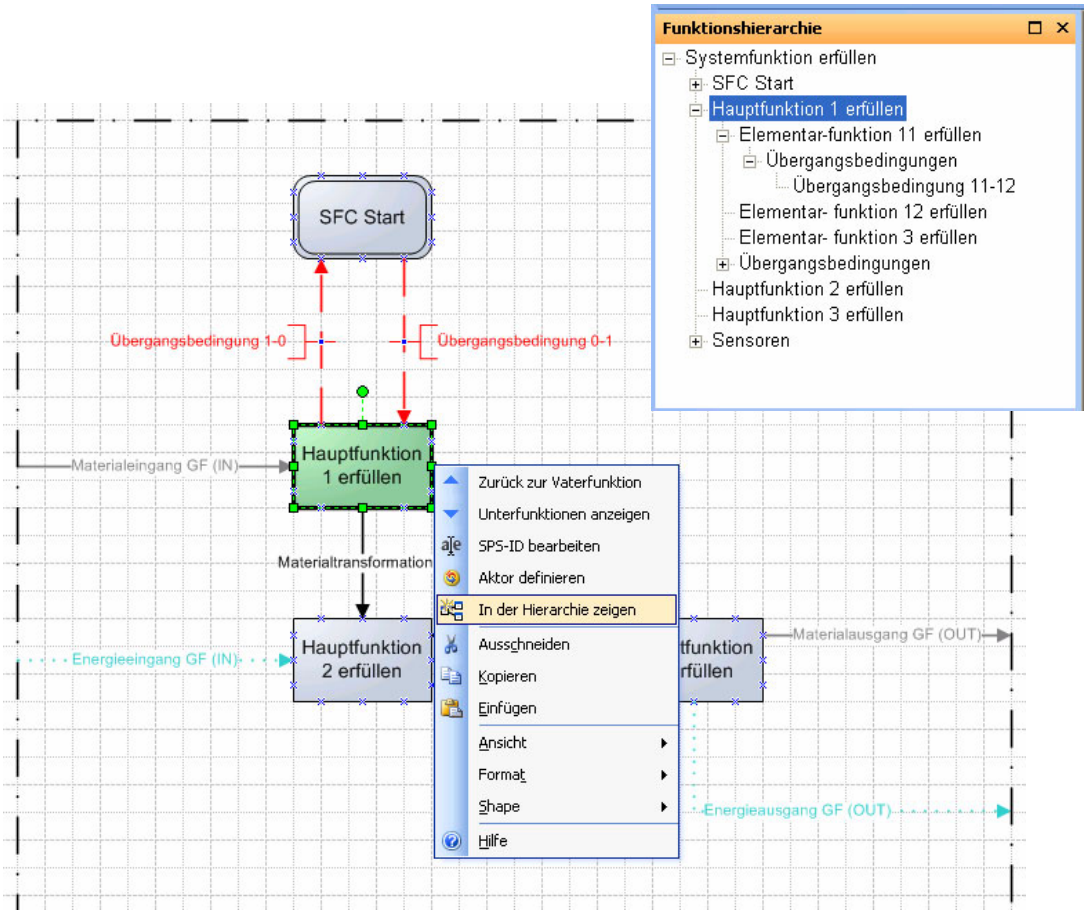


Abbildung 37: Shape in der Hierarchie zeigen

5. ☺-Funktionen

Das Icon ☺ in der Menüleiste beinhaltet zwei ELVAN spezifische Funktionen. Zum einen die Funktion *Neu* und zum anderen die Funktion *Exportieren*. Unter der Funktion *Neu* können gänzlich neue Hauptbaugruppen, Aktoren und Sensoren in ELVAN vordefiniert und hinterlegt werden. Über die Funktion *Exportieren* werden die Arbeitsergebnisse aus der EFS aufbereitetet und in der Form ausgegeben, in welcher sie in die unterschiedlichen Arbeitsumgebungen der Fachdisziplinen importiert werden können. Über diese Funktion werden eine Vorlage für den Morphologischer Kasten, die I/O-Liste und das SFC für die Programmierumgebung *Simatic* und eine erweiterte I/O-Liste für die Simulationsumgebung *WinMOD* generiert. Dazu besteht die Möglichkeit das Byte-Offset für die Outputvariabeln zu bestimmen.

5.1. ☺-Funktion Exportieren

Die Funktion *Exportieren* beinhaltet die vier Unterfunktionen *Morphologischer Kasten, Simatic, WinMOD* und *Byte-Offset ändern*. Sie werden in den folgenden Unterkapiteln 5.1.1. bis 5.1.4. beschrieben. Alle ☺-Exportfunktionen werden wie folgt ausgelöst:

- Menü → ☺ → *Exportieren* → *gewünschte Funktion* (Abbildung 40)

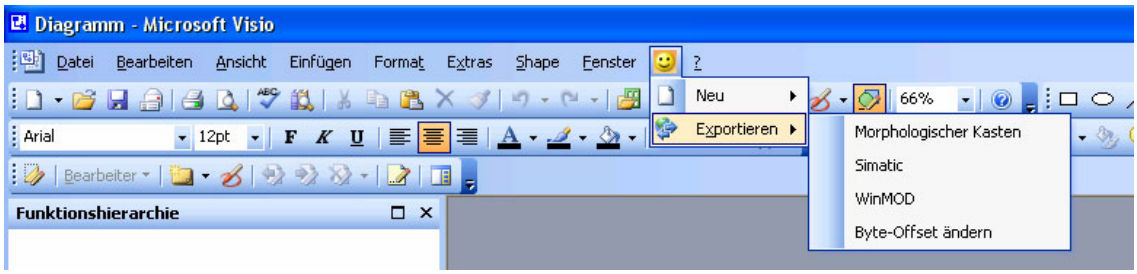


Abbildung 40: ☺-Funktion *Exportieren*

5.1.1. Morphologischer Kasten

Diese Funktion exportiert eine Vorlage zur Erstellung des morphologischen Kastens im MS Excel-Format. Dabei werden alle Elementarfunktionen, die über Flüsse verfügen und welchen noch kein Aktor hinterlegt wurde, abgebildet. Als zusätzliche Information sind alle den entsprechenden Elementarfunktionen zugehörigen Sensoren, Übergangsbedingungen und Flüsse abgebildet und gekennzeichnet. Der Morphologische Kasten wird wie folgt exportiert:

1. Menü → ☺ → *Exportieren* → *Morphologischer Kasten* (Abbildung 40)
2. Darauf erscheint im Projektverzeichnis die MS Excel-Datei *Morphologischer Kasten* (Abbildung 41)

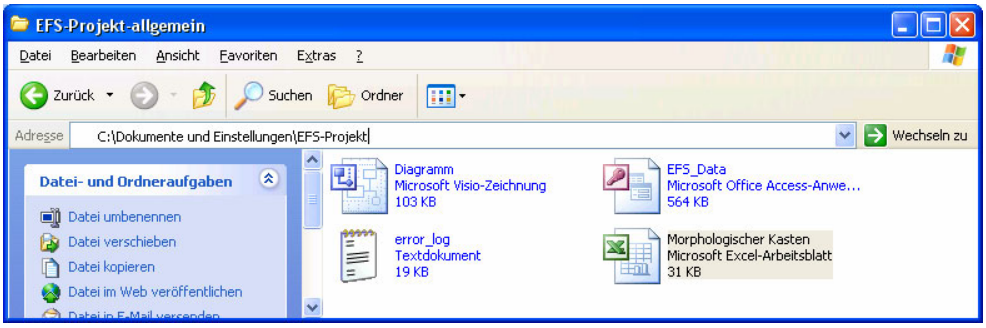


Abbildung 41: MS Excel-Datei *Morphologischer Kasten*

3. Diese Datei kann nun geöffnet werden und dient als Entwurfsvorlage für die Lösungsfindung.

Abbildung 42 zeigt ein Beispiel eines solchen aus ELVAN exportierten Morphologischen Kastens. In der ersten Spalte A sind die Elementarfunktionen und in der zweiten Spalte B die zugehörigen Flüsse, Übergangsbedingungen und Sensoren aufgelistet. In der dritten Spalte C findet sich die Benennung selbiger. Die Richtung des Flusses gibt Aufschluss, ob es sich dabei um einen Funktionseingang oder -ausgang handelt (vgl. Legende). In die leeren Felder können verschiedene Lösungskonzepte eingetragen werden.

	A	B	C	D	E	F	G	H
1		**Morphologischer Kasten**				**Lösungskonzepte**		
2	Unterfunktion 12 erfüllen	←	Materialeingang GF (IN)					
3		←—	42					
4		→	Materialfluss 11					
5		—→	18					
6		⊥	Übergangsbedingung 1-0(OUT)					
7		⊥	Übergangsbedingung 11-12					
8	Hauptfunktion 2 erfüllen	←	Materialfluss 1					
9		⇠	Energieeingang GF (IN)					
10		→	Materialfluss 2					
11		⇢	Energiefluss 1					
12	Hauptfunktion 3 erfüllen	←	Materialfluss 2					
13		⇠	Energiefluss 1					
14		→	Materialausgang GF (OUT)					
15		⇢	Energieausgang GF (OUT)					
16								
17		**Legende**						
18		⊥	Sensor an Funktion					
19		←	Materialfluss					
20		⇠	Energiefluss					
21		←—	Informationsfluss					
22		⊥	Übergangsbedingung					
23		←	Funktionseingang					
24		→	Funktionsausgang					

Abbildung 42: *Morphologischer Kasten* nach dem Export aus ELVAN

5.1.2. Simatic

Diese Funktion exportiert die zwei notwendigen Listen für den Upload des SFC in die Simatic Programmierumgebung im ASCII-Format. Dies sind zum einen die I/O-Liste *symbols* und zum anderen die eigentliche Software *Source*. In der I/O-Liste finden sich sämtliche im Projekt vorkommenden Inputs und Outputs mit ihren genauen Spezifikationen. Die Software beinhaltet neben dem eigentlichen Ablaufprogramm, die simatic-spezifische Konfiguration der SPS. Mit dem Import beider Listen in die Simatic Programmierumgebung ist die SPS fertig programmiert und betriebsbereit. Die beiden Listen werden wie folgt exportiert:

1. Menü → ☺ → *Exportieren* → *Simatic* (Abbildung 40)
2. Darauf erscheinen im Projektverzeichnis die ASCII-Listen *Source* und *symbols* (Abbildung 43)
3. Diese beiden Listen können nun in die Simatic Programmierumgebung importiert werden.

Abbildung 43 zeigt die beiden ASCII-Listen *Source* und *symbols*.

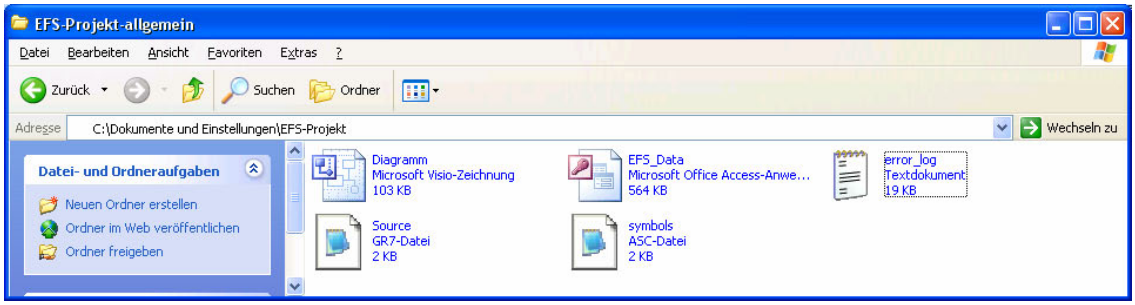


Abbildung 43: ASCII-Listen *Source* und *symbols*

Abbildung 44 zeigt ein Beispiel einer solchen aus ELVAN exportierten I/O-Liste. Die erste und die beiden letzten Zeilen sind simaticspezifisch und bezeichnen die Konfiguration der SPS. In den dazwischen liegenden Zeilen befindet sich die eigentliche I/O-Liste, welche wie folgt aufgebaut ist: In der zweiten Spalte sind die Variabelnamen aufgelistet. In der dritten Spalte wird zwischen Input (I) und Output (Q) unterschieden. In der vierten Spalte findet sich die Angabe des Speicherplatzes und in der fünften Spalte wird die Art der Variabel benannt.

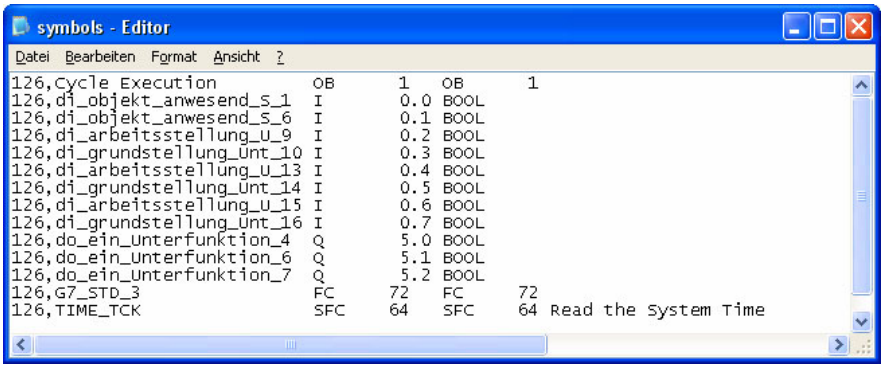


Abbildung 44: I/O-Liste *symbols*

Abbildung 45 zeigt ein Beispiel einer solchen aus ELVAN exportierten SPS-Software. Die ersten drei Abschnitte beinhalten globale Programmeinstellungen. Danach folgt das eigentliche Programm, in dem zuerst die „Steps“ und anschliessend die „Transitionconditions“ mit den zugehörigen Output- und Inputvariabeln definiert sind.

Abbildung 45: SPS-Programm *Source*

5.1.3. WinMOD

Diese Funktion exportiert die erweiterte I/O-Liste *WinMOD_IOS* im MS Excel-Format, welche in die WinMOD Simulationsumgebung importiert werden kann. Sie beinhaltet sämtliche Information, die zur Erstellung der Maschinensimulation in WinMOD notwendig ist. Die erweiterte I/O-Liste wird wie folgt exportiert:

1. Menü → ☺ → *Exportieren* → *WinMOD* (Abbildung 40)
2. Darauf erscheint im Projektverzeichnis die MS Excel-Datei *WinMOD_IOS* (Abbildung 46)
3. Diese Datei kann nun in die WinMOD Simulationsumgebung importiert werden.

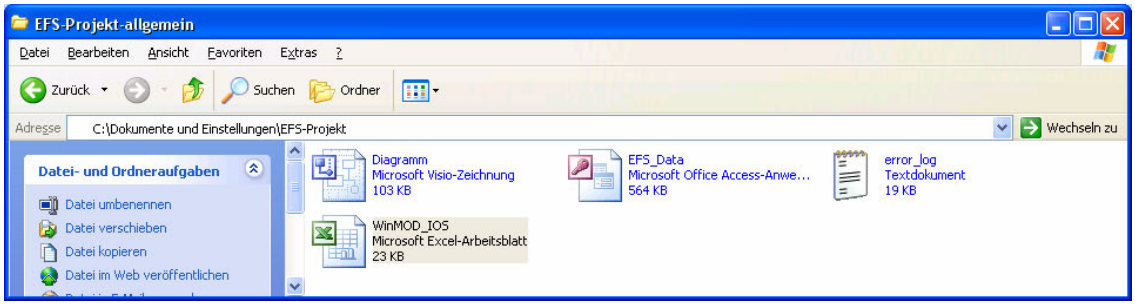
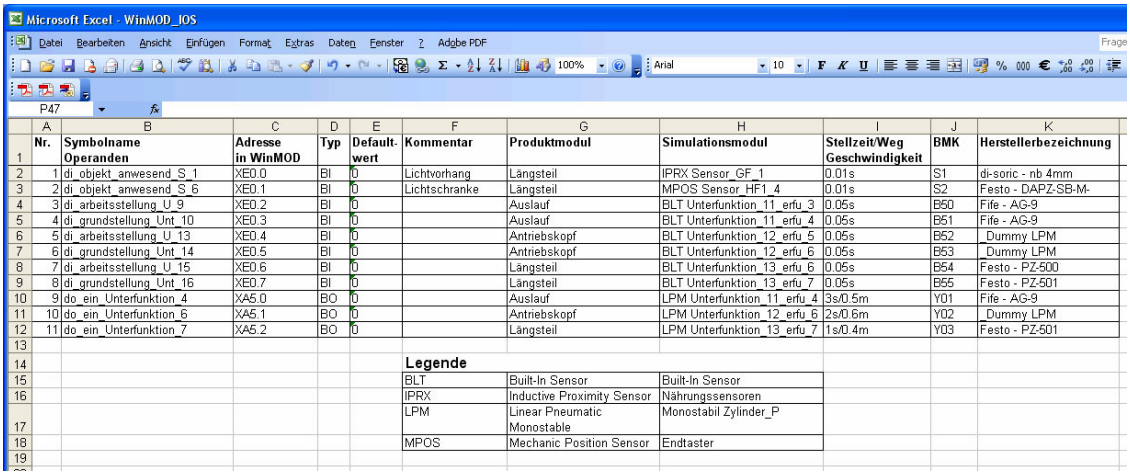


Abbildung 46: MS Excel Liste *WinMOD_IOS*

Abbildung 47 zeigt ein Beispiel einer solchen aus ELVAN exportierten erweiterten I/O-Liste. In der linken Spalte werden die I/O's fortlaufend numeriert. In der folgenden Spalte sind die Variabelnamen aufgelistet und danach folgt die Speicheradresse der Variabel. Die Spalte *Typ* kennzeichnet die Variabelart. Dabei steht *BI* für binärer Input und *BO* für binärer Output. Danach folgt der Variabel-Defaultwert. Unter *Kommentar* wird ein allfälliger persönlicher Kommentar des EFS-Bearbeiters ausgegeben. Die Spalte *Produktmodul* beschreibt die Zugehörigkeit der Variabel auf eine bestimmte Hauptbaugruppe in der EFS. Das *Simulationsmodul* fasst die verschiedenen Inputs und Outputs einer einzelnen Komponente – wie beispielsweise eines bestimmten Aktors – zu einer Gruppe zusammen. In der Spalte *Stellzeit/Weg/Geschwindigkeit* werden die genaueren Angaben des Komponentenverhaltens wiedergegeben. *BMK* steht für das Betriebsmittelkennzeichen der Komponente und die *Herstellerbezeichnung* spezifiziert die verwendete Komponente.



	A	B	C	D	E	F	G	H	I	J	K
1	Nr.	Symbolname Operanden	Adresse in WinMOD	Typ	Default-wert	Kommentar	Produktmodul	Simulationsmodul	Stellzeit/Weg Geschwindigkeit	BMK	Herstellerbezeichnung
2	1	di_objekt_anwesend_S_1	XE0.0	BI	0	Lichtvorhang	Längsteil	IPRX Sensor_GF_1	0.01s	S1	di-soric - nb 4mm
3	2	di_objekt_anwesend_S_6	XE0.1	BI	0	Lichtschranke	Längsteil	MPOS Sensor_HF1_4	0.01s	S2	Festo - DAPZ-SB-M-
4	3	di_arbeitsstellung_U_9	XE0.2	BI	0		Auslauf	BLT Unterfunktion_11_erfu_3	0.05s	B50	Fife - AG-9
5	4	di_grundstellung_Unt_10	XE0.3	BI	0		Auslauf	BLT Unterfunktion_11_erfu_4	0.05s	B51	Fife - AG-9
6	5	di_arbeitsstellung_U_13	XE0.4	BI	0		Antriebskopf	BLT Unterfunktion_12_erfu_5	0.05s	B52	_Dummy LPM
7	6	di_grundstellung_Unt_14	XE0.5	BI	0		Antriebskopf	BLT Unterfunktion_12_erfu_6	0.05s	B53	_Dummy LPM
8	7	di_arbeitsstellung_U_15	XE0.6	BI	0		Längsteil	BLT Unterfunktion_13_erfu_6	0.05s	B54	Festo - PZ-500
9	8	di_grundstellung_Unt_16	XE0.7	BI	0		Längsteil	BLT Unterfunktion_13_erfu_7	0.05s	B55	Festo - PZ-501
10	9	do_ein_Unterfunktion_4	XA5.0	BO	0		Auslauf	LPM Unterfunktion_11_erfu_4	3s/0.5m	Y01	Fife - AG-9
11	10	do_ein_Unterfunktion_6	XA5.1	BO	0		Antriebskopf	LPM Unterfunktion_12_erfu_6	2s/0.6m	Y02	_Dummy LPM
12	11	do_ein_Unterfunktion_7	XA5.2	BO	0		Längsteil	LPM Unterfunktion_13_erfu_7	1s/0.4m	Y03	Festo - PZ-501
13											
14						**Legende**					
15						BLT	Built-In Sensor	Built-In Sensor			
16						IPRX	Inductive Proximity Sensor	Nährungssensoren			
17						LPM	Linear Pneumatic Monostable	Monostabil Zylinder_P			
18						MPOS	Mechanic Position Sensor	Endtaster			
19											

Abbildung 47: Erweiterte I/O-Liste *WinMOD_IOS*

5.1.4. Byte-Offset ändern

Mit dieser Exportfunktion kann das Byte-Offset für die Outputvariabeln gesetzt werden. Sie erleichtert die Konfiguration des Speicherkopplers zum Datenaustausch zwischen Simatic und WinMOD. Das Byte-Offset für die Inputvariabeln ist in ELVAN unveränderlich = 0. Das gewählte Byte-Offset für die Outputvariabeln wird im Simatic- sowie im WinMOD-Export berücksichtigt. Das Byte-Offset für die Outputvariabeln wird wie folgt gesetzt:

1. Menü → ☺ → *Exportieren* → *Byte-Offset ändern* (Abbildung 40)
2. Darauf öffnet sich das Fenster *Byte-Offset ändern* (Abbildung 48)
3. Eintrag *Neuer Offset*: → Die gewünschte Zahl in das Feld eingeben
4. Mit *Ok* bestätigen
   (Abbildung 49 zeigt die beiden I/O-Listen aus dem Simatic- und dem WinMOD-Export mit dem Byte-Offset = 5 für die Outputvariabeln)

Abbildung 48: Fenster Byte-Offset ändern

Abbildung 49 zeigt das angepasste Byte-Offset für die Outputvariabeln in den exportierten I/O-Listen für WinMOD und Simatic.

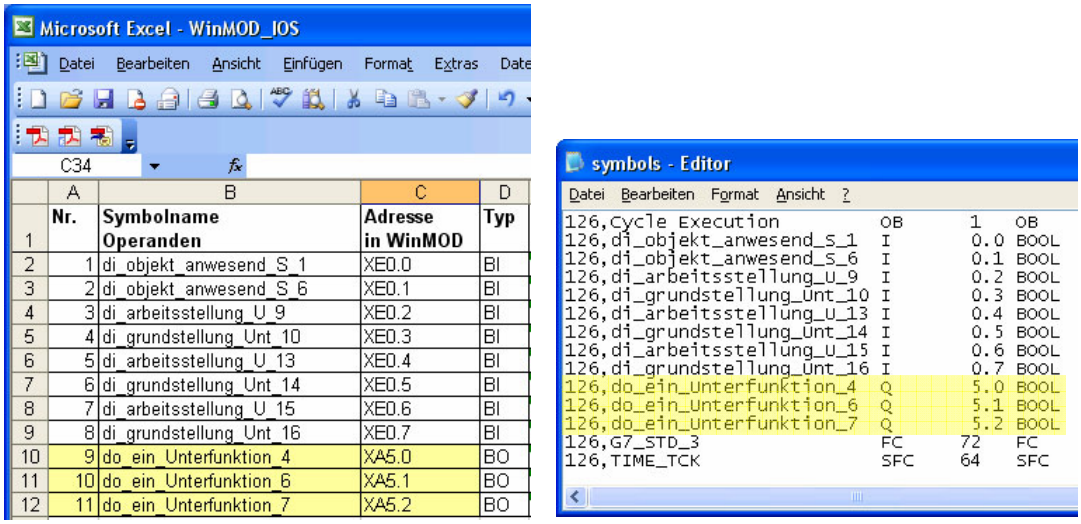


Abbildung 49: Gesetztes Byte-Offset für die Outputvariabeln

Referenzen

[1] Bathelt J., Jönsson A., Bacs C., Dierssen S. und Meier M., „Applying the new VDI design guideline 2206 on mechatronic systems controlled by a PLC“, ICED05 International Conference on Engineering Design, Melbourne, Australien, 2005.

[2] Bathelt J., Frey U., Fankhauser M., Jönsson A., Dierssen S. und Meier M., „A new Guideline to Develop a Lecture – Using the VDI 2221 Frame – Applied on a Mechatronic Course“, ICED05 International Conference on Engineering Design, Melbourne, Australien, 2005.

[3] Bathelt J., Jönsson A., Bacs C., Kunz A. und Meier M., „Conceptual design approach for mechatronic systems controlled by a programmable logic controller (PLC)“, ICED03 International Conference on Engineering Design, Stockholm, Schweden, 2003.

[4] Bathelt J., Jönsson A., Bacs C. und Meier M., „Modularisierung SPS-gesteuerter mechatronischer Systeme“, 1. Paderborner Workshop: Intelligente mechatronische Systeme, veröffentlicht in der HNI Verlagsschriftenreihe, Band 122, Prof. Dr.-Ing. Jürgen Gausemeier (Hrsg.), Rechnerintegrierte Produktion, Bonifatius Verlag, Paderborn, Deutschland, 2003.

[5] Bathelt J. und Jönsson A., „How to Implement the Virtual Machine Concept Using xPC Target“, Nordic MATLAB Conference, Kopenhagen, Dänemark, 2003.

[6] Becker M., „Mechatronischer Entwurf eines reversierenden, hydraulishen Antriebsaktors für die aktive Fahrzeugfederung“, VDI-Verlag, Reihe 12, Nr. 555, Düsseldorf, Deutschland, 2003.

[7] Bernecker + Rainer Industrie Elektronik GmbH, „B&R Automation Studio“, http://www.br-automation.com/, Eggelsberg, Österreich, 2006.

[8] Bonfatti F., Monari P. D. und Sampieri U., „IEC 1131-3 Programming Methodology“, CJ International, Frankreich, 1997.

[9] Bosch Packaging, http://pa.bosch.com/, Deutschland, 2006.

[10] Breiing A. und Flemming M., „Theorie und Methoden des Konstruierens“, Springer Verlag, Berlin/Heidelberg, Deutschland, 1993.

[11] Brütsch Elektronik AG, http://www.brel.ch/, Uhwiesen, Schweiz, 2006.

[12] Buur J. und Andreasen M., „Design models in mechatronic product development“, Journal of Design Studies, Vol.10, Nr.3, 1989.

[13] Buur J., „A Theoretical Approach to Mechatronics Design“, Technical University of Denmark, Dissertation, Dänemark, 1990.

[14] Buur J., „Design models and methods for mechatronics“, Mechatronic Design in Textile Engineering, 27-32, Kluwer Academic Publishers, Niederlande, 1995.

[15] CADsys, „FOD - Function Oriented Design“, Chemnitz, Deutschland, 2006.

[16] Counsell J., Porter I., Dawson D. und Duffy M., „Schemebuilder: computer aided knowledge based design of mechatronic systems“, Journal of Assembly Automation, ISSN: 0144-5154, MCB UP Ltd, Vol. 19, Issue 2, pp. 129-138, 1999.

[17] Dierssen S., „Systemkopplung zur komponentenorientierten Simulation digitaler Produkte“, VDI-Verlag, Reihe 20, Nr. 358, Düsseldorf, Deutschland, 2002.

[18] DIN, „40719-6: Schaltungsunterlagen; Regeln für Funktionspläne“, Beuth Verlag, Berlin, Deutschland, ersetzt durch DIN EN 60848, gültig bis 1.4.2005.

[19] DIN, „40719-11: Schaltungsunterlagen; Zeitablaufdiagramme; Schaltfolgediagramme“, Beuth Verlag, Berlin, Deutschland, August 1978, ersatzlos zurückgezogen.

[20] DIN, „60000: Textilien; Grundbegriffe, Beuth Verlag, Berlin, Deutschland, Januar 1969.“

[21] DIN EN, „60617-12: Graphische Symbole für Schaltpläne - Teil 12: Binäre Elemente“, Ersatz für DIN 40900-12 (1992-09), Beuth Verlag, Berlin, Deutschland, April 1999.

[22] DIN EN, „60848: 2002 GRAFCET, Spezifikationssprache für Funktionspläne der Ablaufsteuerung“, Ersatz für DIN 40719-6, Beuth Verlag, Berlin, Deutschland, Dezember 2002.

[23] DIN EN, „61131-3: Speicherprogrammierbare Steuerungen Teil 3: Programmiersprachen“, Beuth Verlag, Berlin, Deutschland, Dezember 2003.

[24] DIN EN, „61131-3 Beiblatt 1: Speicherprogrammierbare Steuerungen - Leitlinien für die Anwendung und Implementierung von Programmiersprachen für Speicherprogrammierbare Steuerungen“, Beuth Verlag, Berlin, Deutschland, April 2005.

[25] Eisenhut A., „Service Driven Design : Konzepte und Hilfsmittel zur informationstechnischen Kopplung von Service und Entwicklung auf der Basis moderner Kommunikations-Technologien“, VDI-Verlag, Reihe 20, Nr. 297, Deutschland, 1999.

[26] FESTO, „Fluiddraw“, http://www.fluiddraw.de, Deutschland, 2006.

[27] Flath M., „Methode zur Konzipierung mechatronischer Produkte“, HNI Verlagsschriftenreihe, Band 108, Prof. Dr.-Ing. Jürgen Gausemeier (Hrsg.), Rechnerintegrierte Produktion, Bonifatius Verlag, Paderborn, Deutschland, 2002.

[28] GEPA, „VersionWorks for Automation“, http://www.versionworks.de/, Landau, Deutschland, 2006.

[29] Gritec Institut für angewandte Technologie AG, http://www.gritec.ch/, Schiers, Schweiz, 2006.

[30] Haberfellner R., Nagel P., Becker M., Büchel A. und von Massow H., „Systems engineering: Methodik und Praxis“, Verlag Industrielle Organisation, Hrsg.: W. F. Daenzer / F. Huber, 11. durchgesehene Aufl., Zürich, Schweiz, 2002.

[31] Hahn M., „OMD - Ein Objektmodell für den Mechatronikentwurf“, VDI-Verlag, Reihe 20, Nr. 299, Düsseldorf, Deutschland, 1999.

[32] Hahn M., Lückel J. und Wittler G., „Eine Entwurfsmethodik für mechatronische Systeme“, Tagungsband zu den 3. Magdeburger Maschinenbautagen, Band.2, Seite 1-12, Magdeburg, Deutschland, 1997.

[33] Intelliact, „Intelliact Mediator“, http://www.intelliact.ch, Zürich, Schweiz, 2006.

[34] ISO, „11898-1: Road vehicles - Controller area network - Part 1: Data link layer and physical signalling, Beuth Verlag, Berlin, Deutschland, Dezember 2003.

[35] ISO, „8159: Textiles - Morphology of fibres and yarns - Vocabulary“, Beuth Verlag, Berlin, Deutschland, April 1987.

[36] iXtronics, „CAMeL-View TestRig“, http://www.ixtronics.com/, Paderborn, Deutschland, 2006.

[37] Jeckle M., Rupp C., Hahn J., Zengler B. und Queins S., „UML 2 glasklar“, Carl Hanser Verlag München Wien, Deutschland, 2004.

[38] Jönsson A., Bathelt J. und Broman G., „Implications of modelling one-dimensional impact by using a spring and damper element“, Proceedings of the I MECH E Part K Journal of Multi-body Dynamics, Vol. 219, No. 7, Professional Engineering Publishing, London, Oktober, 2005.

[39] Jönsson A., Bathelt J. und Broman G., „Interacting with real time simulations – virtual reality in industry applications“, IPT-EGVE, Zürich, Schweiz, 2003.

[40] Jönsson A., „Lean Prototyping of Multi-body and Mechatronic Systems“, Dissertation Series No. 2004:08, Blekinge Institute of Technology, Karlskrona, Schweden, 2004.

[41] Kallenbach E., Saffert E., Schäffel C. und Birli O., „Zur Gestaltung integrierter mechatronischer Produkte“, Tagung Mechatronik im Maschinen- und Fahrzeugbau, 10. - 12. März 1997 , Moers, VDI Berichte 1315, VDI-Verlag, Düsseldorf, Deutschland, 1997.

[42] Kallenbach E., Zöppig V., Birli O., Feindt K., Ströhla T., Saffert E. und Schmidt J., „Integration mechatronischer Systeme“, 4. VDI Mechatronik Tagung 2001 Innovative Produktentwicklungen, Frankenthal, 12. und 13. September, VDI-Berichte 1631, VDI-Verlag, Düsseldorf, Deutschland, 2001.

[43] Kallmeyer F., „Eine Methode zur Modellierung prinzipieller Lösungen mechatronischer Systeme“, HNI Verlagsschriftenreihe, Band 42, Prof. Dr.-Ing. Jürgen Gausemeier (Hrsg.), Rechnerintegrierte Produktion, Bonifatius Verlag, Paderborn, Deutschland, 1998.

[44] Köckerling M., „Methodische Entwicklung und Optimierung der Wirkstruktur mechatronischer Produkte“, HNI Verlagsschriftenreihe, Band 143, Prof. Dr.-Ing. Jürgen Gausemeier (Hrsg.), Rechnerintegrierte Produktion, Bonifatius Verlag, Paderborn, Deutschland, 2004.

[45] Kreusch K., „Verifikation numerischer Steuerungen an virtuellen Werkzeugmaschinen“, Shaker Verlag, Aachen, Deutschland, 2002.

[46] KTI/CTI Förderagentur für Innovation des Bundesamtes für Berufsbildung und Technologie, http://www.bbt.admin.ch/kti/profil/d/, Bern, Schweiz, 2006.

[47] Leumann A. und Wieland F., „Kommerzialisierbarkeit von Methoden und Tools“, Semesterarbeit, Zentrum für Produkt-Entwicklung, ETH Zürich, Schweiz, 2005.

[48] Meier M., Dierssen S. und Bathelt J., „Die Virtuelle Maschine - oder wie Inbetriebnahme vor Montage möglich ist“, CPK 2004, 4. Chemnitzer Produktionstechnisches Kolloquium, Chemnitz, Deutschland, 2004.

[49] Meyer U., Creux S. und Marin A., „Grafische Methoden der Prozessanalyse“, Carl Hanser Verlag, Deutschland, 2005.

[50] Mewes & Partner GmbH, http://www.winmod.de/, 2006.

[51] Microsoft, „Visual SourceSafe “, http://msdn.microsoft.com/ssafe/, 2006.

[52] Microsoft, „Visio“, http://office.microsoft.com/visio/, 2006.

[53] Molt T., „Eine domänenübergreifende Softwarespezifikationstechnik für automatisierte Anlagen“, HNI Verlagsschriftenreihe, Band 121, Prof. Dr.-Ing. Jürgen Gausemeier (Hrsg.), Rechnerintegrierte Produktion, Bonifatius Verlag, Paderborn, Deutschland, 2003.

[54] NASKITA, „Graftor - Grafcet editor“, http://www.naskita.com/, 2006.

[55] Nagy Z., Porter I. und Bradshaw A., „Schemebuilder: cross-domain modelling example“, International Journal of Electrical Engineering Education, ISSN: 0020-7209, Vol. 41, Issue 4, pp. 341-349, Oktober 2004.

[56] OMG - Object Management Group, „Unified Modeling Language (UML)“, version 2.0, formal/05-07-04, http://www.uml.org/, Needham, USA, August 2005.

[57] Osmers, U., „Projektieren speicherprogrammierbarer Steuerungen mit Virtual Reality“, Forschungsberichte aus dem Institut für Werkzeugmaschinen und Betriebstechnik der Universität Karlsruhe, Band 87, Karlsruhe, Deutschland, 1998.

[58] Otto K. N., „A Process for Modularizing Product Families“, ICED01 International Conference on Engineering Design, Glasgow, England, 2001.

[59] Pahl G. und Beitz W., „Engineering design: a systematic approach“, Springer, Berlin, Deutschland, 1999.

[60] POA - Process Oriented Analysis, http://www.poa.texma.org/Toolbox/toolbox.html/, 2006.

[61] „Rieter Textile Systems“, http://www.rieter.com/main/textile/, Winterthur, Schweiz, 2006.

[62] Schätz B., Fahrmair M., von der Beeck M., Jack P., Kespohl H., Koc A., Liccardi B., Scheermesser S. und Zündorf A., "Entwicklung, Produktion und Service von Software für eingebettete Systeme in der Produktion", Abschlussbericht der Vordringlichen Aktion des Bundesministeriums für Bildung und Forschung, Deutschland, Februar 2000.

[63] Schön A., „Konzept und Architektur eines Assistenzsystems für die mechatronische Produktentwicklung“, Dissertation, Lehrstuhl für Konstruktionsmethodik der Friedrich-Alexander-Universität Erlangen-Nürnberg, Deutschland, 2000.

[64] SIEMENS, „CamTool“, Projektierungshandbuch, Bestellnummer: 6AU1900-1AB32-0AA0, Deutschland, Dezember 2004.

[65] SIEMENS, „SIMATIC“, http://www.automation.siemens.com/simatic, Nürnberg-Moorenbrunn, Deutschland, 2006.

[66] SIEMENS, „SIMIT“, http://www.siemens.de/simit, Erlangen, Deutschland, 2006.

[67] Stone R., „Towards a Theory of Modular Design“, Dissertation, University of Texas at Austin, USA, 1997.

[68] Toepper S., Lückel J., Moritz W., Kuhlbusch W. und Scharfeld F., „Modellgestützter Entwurf des Parallelroboters TRIPLANAR“, 4. VDI Mechatronik Tagung 2001 Innovative Produktentwicklungen, Frankenthal, 12. und 13. September, VDI-Berichte 1631, VDI-Verlag, Düsseldorf, Deutschland, 2001.

[69] UGS, „eM-PLC“, http://www.ugs.com/products/tecnomatix/production_execution/em_plc.shtml, Plano, Texas, USA, 2006.

[70] UGS, „Teamcenter“, http://www.ugs.com/products/teamcenter/, Plano, Texas, USA, 2006.

[71] UGS, „NX“, http://www.ugs.com/products/nx/, Plano, Texas, USA, 2006.

[72] VDI, „2206: Entwicklungsmethodik für mechatronische Systeme“, Beuth Verlag, Berlin, Deutschland, Juni 2004.

[73] VDI, „2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte“, Beuth Verlag, Berlin, Deutschland, Mai 1987.

[74] VDI, „2803: Funktionenanalyse - Grundlagen und Methode“, Beuth Verlag, Berlin, Deutschland, Oktober 1996.

[75] VDI, „3260: Funktionsdiagramme von Arbeitsmaschinen und Fertigungsanlagen“, Beuth Verlag, Berlin, Deutschland, Dezember 1994, ersatzlos zurückgezogen, der VDI empfiehlt die Anwendung von DIN 40719-6.

[76] VDI/VDE, „2422: Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik“, Beuth Verlag, Berlin, Deutschland, Februar 1994.

[77] VDI/VDE, „3684: Herstellerneutrale Konfiguration von Antriebssystemen - Beschreibung ereignisgesteuerter Bewegungsabläufe mit Funktionsplänen“, Beuth Verlag, Berlin, Deutschland, September 1997.

[78] Wang L., Shen W., Xie H., Neelamkavil J. und Pardasani A., „Collaborative conceptual design - state of the art and future trends“, Journal of Computer-Aided Design, Vol. 24, pp. 981-996, Elsevier Science Ltd (Hrsg.), 2002.

[79] Weck Manfred, „Werkzeugmaschinen - Fertigungssysteme“, Band 3.1, Automatisierung und Steuerungstechnik 1, Vierte grundlegend neu bearbeitete und erweiterte Auflage, VDI Verlag, Düsseldorf, Deutschland, 1995.

[80] Werner Götz, „Hydraulik in Theorie und Praxis“, Robert Bosch GmbH Geschäftsbereich Automationstechnik Schulung, Omega Fachliteratur, Ditzingen, Deutschland, 1997.

Lebenslauf

Name	Jens Bathelt
Geburtsdatum	3. Juni 1968
Nationalität	deutsch
Zivilstand	geschieden
Kinder	Björn *1991, Ronja *1993

Sep. 1984 - Juli 1987	UHDE GmbH (Hoechst), Dortmund D Ausbildung zum Technischen Zeichner
Juli 1987 - Aug. 1987	UHDE GmbH (Hoechst), Dortmund D Technischer Zeichner
Aug. 1987 - Juli 1988	Fachoberschule für Technik, Dortmund D Fachhochschulreife in der Fachrichtung Maschinentechnik
Sep. 1988 - Juni 1989	DDS ENGINEERING - DANISCO, Kopenhagen DK Technischer Zeichner
Sep. 1989 - Sep. 1992	Gerhard-Mercator-Universität, Duisburg D Fachgebundene Hochschulreife
Nov. 1993 - Dez. 1993	Institut für Mechatronik, IMECH, Moers D Studentische Hilfskraft
Okt. 1989 - Sep. 1995	Gerhard-Mercator-Universität, Duisburg D Diplom Technomathematik
Sep. 1995 - Juli 2001	Precisionsoft AG, Au/ZH CH CAD Softwareentwickler
Juni 2003 - Sep. 2003	Blekinge Tekniska Högskola (BTH), Karlskrona SE Wissenschaftlicher Mitarbeiter
seit Feb. 2001	Eidgenössische Technische Hochschule (ETH) Zürich, Zürich CH Doktorand am ZPE

Zürich, im Oktober 2006 *Jens Bathelt*